# EFFICIENT ESTIMATION OF KERNEL SURROGATE MODELS FOR TASK ATTRIBUTION

**Zhenshuo Zhang*** Northeastern University
**Minxuan Duan*** Northeastern University
**Hongyang R. Zhang*** Northeastern University

## ABSTRACT

Modern AI agents such as large language models are trained on diverse tasks—translation, code generation, mathematical reasoning, and text prediction—simultaneously. A key question is to quantify how each individual training task influences performance on a target task, a problem we refer to as *task attribution*. The direct approach, leave-one-out retraining, measures the effect of removing each task, but is computationally infeasible at scale. An alternative approach that builds surrogate models to predict a target task's performance for any subset of training tasks has emerged in recent literature. Prior work focuses on linear surrogate models, which capture first-order relationships, but miss nonlinear interactions such as synergy, antagonism, or XOR-type effects. In this paper, we first consider a unified task weighting framework for analyzing task attribution methods, and show a new connection between linear surrogate models and influence functions through a second-order analysis. Then, we introduce *kernel surrogate models*, which more effectively represent second-order task interactions. To efficiently learn the kernel surrogate, we develop a gradient-based estimation procedure that leverages a first-order approximation of pretrained models; empirically, this yields accurate surrogate estimates with less than $2\%$ relative error without repeated retraining. Experiments across multiple domains—including mathematical reasoning in transformers, in-context learning, and multi-objective reinforcement learning—demonstrate the effectiveness of kernel surrogate models. They achieve a $25\%$ higher correlation with the leave-one-out ground truth than linear surrogates and influence-function baselines, enabling more accurate and scalable task attribution. When used for downstream task selection, kernel surrogate models further yield a $40\%$ improvement in demonstration selection for in-context learning and multi-objective reinforcement learning benchmarks.

## 1 INTRODUCTION

Modern AI systems are trained to perform well across a diverse set of tasks, ranging from machine translation and code generation to object recognition and mathematical reasoning. This broad, multi-task capability raises a central question in interpretability: *how do individual training tasks contribute to model performance*?

In this work, we study *task attribution*, the problem of quantifying the influence of each training task on downstream behavior. Beyond being conceptually fundamental, task attribution has direct implications across several application domains. In multi-task learning, understanding task relationships can guide the design of neural architectures and loss-reweighting strategies (Wu et al., 2020; Yang et al., 2025). In multi-group learning (Deng & Hsu, 2024), it can reveal how training on different demographic groups shapes model behavior. In in-context learning (Garg et al., 2022; Zhang et al., 2025), it parallels the question of how adding or removing a single demonstration affects predictions Min et al. (2022). Related questions also arise in multi-objective reinforcement learning (Yu et al., 2020; Zhang et al., 2026), where one wants to understand how competing reward signals influence learned policy. Accurate task attribution can also help understand when the representations generated by pre-trained large-scale models can be used in downstream tasks (Wu et al., 2023).

*Email Address: {zhang.zhens, duan.mi, ho.zhang}@northeastern.edu.

A natural estimator of task influence is leave-one-out (LOO) retraining, which measures the change in performance when one task is withheld from training. However, if the training set contains $K$ tasks, computing LOO scores requires $K + 1$ full training runs, which is prohibitive. More efficient proxies for the LOO scores use influence functions (Koh & Liang, 2017; Grosse et al., 2023), which estimate how model predictions would change if a training sample were added or removed. These methods require evaluating Hessian-vector products, and are usually applied at the terminal state once the model has memorized the training data (Brown et al., 2021). However, computing influence scores for all $K$ tasks still requires repeatedly Hessian approximations (Basu et al., 2021; Kwon et al., 2024). A complementary line of work—datamodels (Ilyas et al., 2022) and data attribution (Park et al., 2023; Bae et al., 2024)—builds surrogate functions that approximate black-box model behaviors. These methods treat the training pipeline as a function mapping data (or task) subsets to predictions, and learn a separate surrogate to emulate the mapping. Once trained, the surrogate provides a faster mechanism for evaluating how subset combinations contribute to a given test distribution. Recent work has primarily focused on *linear surrogate models*, which enjoy appealing theoretical properties, as they can capture first-order effects and admit linear sampling complexities in multi-task settings (Li et al., 2023; 2024a; Zhang et al., 2025). However, linear surrogates fail to capture second-order interactions where task effects are non-additive (See illustration in Figure 2).

In this paper, we revisit the surrogate modeling objective through a unified task-weighting framework. We begin by analyzing linear surrogate models using a second-order Taylor expansion in the weight space. We find that linear surrogate models estimated via minimizing the surrogate modeling objective are approximately equal to the influence functions, up to third-order expansion errors (Proposition 3.1). In particular, when the second-term expansion error is small, then linear surrogates and influence functions are approximately the same (Corollary 3.2). A key technical tool involves applying the delta method to certain covariance statistics in the surrogate regression.

Motivated by these observations, we propose *kernel surrogate models* (KERNELSM), which extend surrogate modeling beyond the additive structure inherent in linear methods. Kernel surrogates—such as those based on radial basis function (RBF) kernels—naturally capture geometric relationships in the 0-1 subset space. A direct kernel estimation, however, would require computing model outcomes on many task subsets. Instead, we design an efficient estimation that uses *gradients as features*, leveraging a first-order approximation of model outputs in its parameters. Empirically, we find that the accuracy of this first-order approximation is under $1\%$ relative error across diverse datasets, including CIFAR-10, modular arithmetic, in-context learning, and multi-objective RL benchmarks, on models ranging from small MLP classifiers to language models with up to 34B parameters. Theoretically, we bound the error of this estimation assuming the errors are small (Proposition A.2).

We evaluate the kernel surrogate modeling framework across a broad range of datasets and architectures. On modular arithmetic reasoning tasks—where the input-output mapping can be XOR, division, quadratic—kernel surrogates improve attribution accuracy by up to $42\%$ relative to existing approaches (Li et al., 2023; Ilyas et al., 2022; Park et al., 2023). We also observe similar gains on in-context learning and sequential decision-making tasks. Applied to the Qwen3-8B model on sentiment classification and mathematical reasoning, kernel surrogates improve attribution accuracy by $18\%$ over prior methods. In reinforcement learning with soft actor-critic on the Meta-World MT10 benchmark (Yu et al., 2020), our method provides $5\%$ improvement on environments with dynamically shifting data distributions. Overall, kernel surrogates consistently outperform prior methods across all settings, increasing correlation with leave-one-out estimates by $25\%$ relative to existing attribution methods. Finally, for downstream task selection, our approach yields $40\%$ lower loss in both LLM inference (demonstration selection for in-context learning) and multi-objective optimization on Meta-World, while using runtime comparable to fitting linear surrogate models.

Taken together, this paper provides an *efficient estimation algorithm for the task-attribution problem via kernel surrogate models*. First, we analyze linear surrogate models, showing that influence functions and the estimated regression coefficients are approximately the same under certain assumptions. Second, we introduce KERNELSM, along with an efficient gradient-based estimation via kernel ridge regression. Thus, our approach is both scalable and flexible for modeling nonlinear task relationships in black-box systems. Third, we evaluate KERNELSM in a variety of empirical settings including in-context learning, mathematical reasoning, and reinforcement learning. We provide the code to replicate our empirical results at https://github.com/VirtuosoResearch/Kernel-surrogate-models.

## 2 PRELIMINARIES

We begin by introducing the notation and a task weighting framework. Within this framework, we provide a formal definition of *task attribution*. We then present influence functions and task modeling techniques from the data attribution literature within this unified framework, which also serve as state-of-the-art baselines for evaluating task attribution.

**Problem setup.** Suppose the training dataset consists of $K$ tasks, denoted by $T_1, T_2, \ldots, T_K$. Each task $k$ contains $n_k$ samples drawn from its task-specific distribution, i.e., $T_k = \{(x_{k,j}, y_{k,j})\}_{j=1}^{n_k}$, for $k = 1, \ldots, K$. The total number of training samples is equal to $n = \sum_{k=1}^{K} n_k$.

We consider a model $f_W$ such as a multitask network parameterized by $W \in \mathbb{R}^d$, which is shared across all tasks. For each task $k$, we denote the task-level loss by $\ell_k(f_W, T_k)$. To specify which tasks are included in training, we introduce a $K$-dimensional binary vector $\mathbf{s} = [s_1, \cdots, s_K]^\top$, where $s_k = 1$ indicates that task $k$ is selected in $\mathbf{s}$ and $s_k = 0$ otherwise. The corresponding weighted empirical loss is then defined as

$$\hat{L}(f_W, \mathbf{s}) = \left( \sum_{j=1}^{K} s_j \right)^{-1} \sum_{k=1}^{K} s_k \ell_k(f_W, T_k). \tag{1}$$

For example, when $s_k = 1$ for all $k$, the weighted loss $\hat{L}(f_W, \mathbf{s})$ reduces to the average loss over all tasks. We denote by $\widehat{W}(\mathbf{s}) \leftarrow \arg\min_W \hat{L}(f_W, \mathbf{s})$ the minimizer of the weighted loss, and by $f_{\widehat{W}(\mathbf{s})}$ the corresponding model.

**Influence estimation.** We aim to evaluate the influence of training tasks on a target test task $T_{\text{test}}$. To this end, we define a performance metric $F(\mathbf{s}) = \mathbb{E}\left[\ell_{\text{test}}(f_{\widehat{W}(\mathbf{s})}, T_{\text{test}})\right]$, which measures the test loss of the model trained with subset of training tasks indicated by $\mathbf{s}$.

A natural way to quantify the contribution of a task is through leave-one-out (LOO) retraining, which measures the change in performance when task $k$ is excluded from the training set

$$\mathcal{I}_k^{\text{LOO}} = F\big([\mathbf{1}]_K\big) - F\big([\mathbf{1}]_K - e_k\big),$$

where $[\mathbf{1}]_K \in \mathbb{R}^K$ is the all-ones vector corresponding to training on all tasks, and $e_k \in \{0,1\}^K$ is the basis vector corresponding to the $k$-th coordinate.

**Definition 2.1.** *The ground-truth task attribution score of task $k$ is given by $\mathcal{I}_k^{LOO}$, for $k = 1, \ldots, K$.*

The other approach for capturing task influence is by computing influence functions on the weighted loss. Given $\mathbf{s}$, the influence of task $k$ on $F(\mathbf{s})$ is formally given by (Koh & Liang, 2017)

$$\mathcal{I}_k(\mathbf{s}) = [\nabla_W F(\mathbf{s})]^\top \left[\nabla_W^2 \hat{L}(f_W, \mathbf{s})\right]^{-1} \frac{s_k \nabla_W \ell_k(f_W, T_k)}{\sum_{j=1}^{K} s_j}. \tag{2}$$

In words, $\mathcal{I}_k(\mathbf{s})$ quantifies how $\hat{L}$ changes when task $k$ is infinitesimally up-weighted. The gradient captures task $k$'s direct contribution, and the Hessian inverse accounts for curvature of the loss. The outer product with $\nabla_W F(\mathbf{s})$ maps the perturbation to its effect on $\hat{L}$. We provide the derivation of equation (2) in Appendix A.

**Linear surrogate models.** An alternative approach to quantify task influence is through task modeling (Li et al., 2023). The key idea is to sample binary vectors $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(m)}$ from $\{0,1\}^K$, each indicating a random subset of tasks. For each $\mathbf{s}^{(j)}$, we train the model on the corresponding weighted loss to obtain $\widehat{W}(\mathbf{s}^{(j)})$ and record its performance $F(\mathbf{s}^{(j)})$. We then fit a linear surrogate model, parameterized by intercept $\alpha \in \mathbb{R}$ and coefficients $\beta = [\beta_1, \ldots, \beta_K]^\top \in \mathbb{R}^K$, by minimizing

$$R(\alpha, \beta) = \mathbb{E}_{\mathbf{s} \sim \mathcal{D}}\left[\left(F(\mathbf{s}) - \alpha - \beta^\top \mathbf{s}\right)^2\right], \tag{3}$$

where $\mathcal{D}$ is a distribution of $\mathbf{s}$ such as the Binomial distribution. The optimal linear surrogate model is obtained by minimizing the empirical loss $\hat{R}(\alpha, \beta)$ on $m$ sampled pairs $\{(\mathbf{s}^{(j)}, F(\mathbf{s}^{(j)}))\}_{j=1}^{m}$.

# 3 METHODS

In this section, we present a kernel-based surrogate modeling approach for influence estimation. First, we analyze surrogate modeling together with influence functions using a second-order Taylor expansion in the task weighting space. We develop a second-order approximation of the linear surrogate model coefficients and find that it is approximately equal to the influence function. Specifically, linear surrogate models provide an estimation of the influence functions if second-order task interactions are nearly zero in the composition. Second, we design a kernel-based surrogate modeling procedure to learn the nonlinear interactions between tasks. Finally, we introduce an efficient gradient estimation algorithm to efficiently learn the kernel surrogate.

## 3.1 ANALYZING SURROGATE MODELING IN THE ATTRIBUTION SPACE

**Second-order analysis of linear surrogate modeling.** We center our analysis around $\mathbf{s}^\star = [\mathbf{1}]_K$, which corresponds to the uniform weight applied to all tasks. Then, we characterize task contributions with respect to this global model. Expanding around the anchor point $\mathbf{s}^*$, we approximate the performance function $F(\mathbf{s})$ using a second-order Taylor expansion as follows:

$$F(\mathbf{s}) \approx F(\mathbf{s}^\star) + \underbrace{[\nabla_{\mathbf{s}} F(\mathbf{s}^\star)]^\top (\mathbf{s} - \mathbf{s}^\star)}_{\text{First-order linear effects}} + \underbrace{\frac{1}{2}(\mathbf{s} - \mathbf{s}^\star)^\top \mathbf{H}_{\mathbf{s}} (\mathbf{s} - \mathbf{s}^\star)}_{\text{Second-order interaction effects}}, \tag{4}$$

where the Hessian matrix $\mathbf{H}_{\mathbf{s}} = \nabla^2_{\mathbf{s}} F(\mathbf{s}^\star)$ captures the nonlinear interactions between tasks.

We next connect this expansion to linear task modeling, which approximates $F(\mathbf{s})$ with linear surrogates of the form $\alpha + \beta^\top \mathbf{s}$. To analyze the behavior of linear surrogates, we apply *a second-order expansion of $F(\mathbf{s})$ in equation (4) into the surrogate modeling objective, $\hat{R}(\alpha, \beta)$.* The following proposition formalizes the resulting characterization of the linear regression coefficients from minimizing $\hat{R}(\alpha, \beta)$ to get the regression coefficients $\hat{\alpha}, \hat{\beta}$.

**Proposition 3.1.** *Let $F : \{0,1\}^K \to \mathbb{R}$ and $\mathbf{s}^\star = [\mathbf{1}]_K$. Assume the third (partial) derivatives of $F(\cdot)$ are bounded by $c_3$ for some small enough $c_3 > 0$. Suppose each coordinate in $\mathbf{s}$ is drawn independently from a Bernoulli distribution with probability $p$ for some fixed $p \in (0,1)$. Let $\hat{\alpha}, \hat{\beta}$ be the minimizer of the linear surrogate objective $\hat{R}(\alpha, \beta)$ on $m$ random samples $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)}$. Then, with probability $1 - \delta$ over the randomness of $m$ sampled subsets, we have*

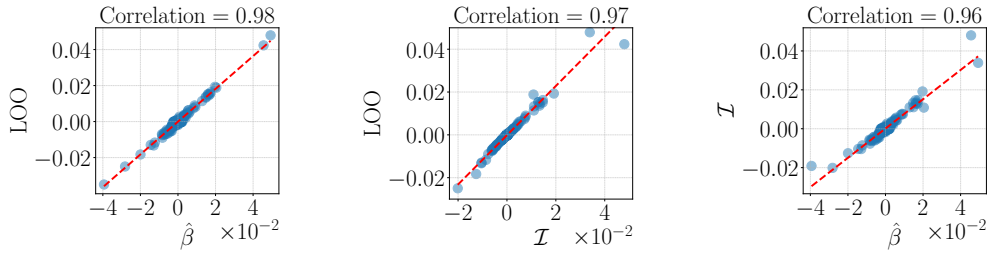
$$\left\| \hat{\beta} - \nabla_{\mathbf{s}} F(\mathbf{s}^\star) - (p-1)\mathbf{H}_{\mathbf{s}}[\mathbf{1}]_K - \tfrac{1-2p}{2} \operatorname{diag}[\mathbf{H}_{\mathbf{s}}] \right\| \lesssim c_3 K^{\frac{3}{2}} p^{-1} + \sqrt{\frac{K + \log(\delta^{-1})}{m}}. \tag{5}$$

This result shows that the surrogate coefficients $\hat{\beta}$ recover the gradient of $F(\mathbf{s}^\star)$ with respect to the weights up to two terms induced by the Binomial distribution. The first corresponds to a sampling shift due to the mean of the Bernoulli draw, while the second reflects the variance of task inclusion in the random subsets. The key idea is based on the *second-order delta method applied to certain certain covariance statistics*. A detailed proof of this proposition is in Appendix A.1.

**Connection between linear surrogate models and influence functions.** This result also shows the connection between the linear task modeling and influence functions. Specifically, under a first-order approximation ($\|\mathbf{H}\|_2 \leq c_2$), both methods estimate the same underlying quantity $\nabla_{\mathbf{s}} F(\mathbf{s}^\star)$. Under this assumption, the second-order bias terms in equation (5) vanish, showing that the linear task modeling coefficients approximate the gradient of the performance landscape: $\hat{\beta} \approx \nabla_{\mathbf{s}} F(\mathbf{s}^\star)$. Meanwhile, one can show that the influence function computes $\vec{\mathcal{I}} = [\mathcal{I}_1(\mathbf{s}^\star), \mathcal{I}_2(\mathbf{s}^\star), \ldots, \mathcal{I}_K(\mathbf{s}^\star)]^\top = \nabla_{\mathbf{s}} F(\mathbf{s}^\star)$. We summarize this discussion precisely as follows.

**Corollary 3.2.** *In the setting of Proposition 3.1, assume further that the second-order task interactions are near zero, i.e., the spectral norm of $\|\mathbf{H}_{\mathbf{s}}\|_2 \leq c_2$ for some small $c_2 > 0$, the linear surrogate model coefficients satisfy that with probability at least $1 - \delta$ over the randomness of $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)}$ for any $\delta > 0$:*

$$\left\| \hat{\beta} - \vec{\mathcal{I}} \right\| \leq c_2(\sqrt{K} + 1/2) + O\left(c_3 K^{3/2} p^{-1}\right) + O\left(\sqrt{\frac{K + \log(\delta^{-1})}{m}}\right). \tag{6}$$




(a) LOO / Linear Surrogate (b) LOO / IF (c) IF / Linear Surrogate

Figure 1: We compare influence functions (IF), leave-one-out (LOO) retraining, and linear surrogate models. Each point corresponds to the individual effect of removing one training example.

Thus, provided that $c_2, c_3$ are small enough and $m \geq O(K + \log(\delta^{-1}))$, then influence functions can approximate linear surrogate models, while linear surrogate models capture additive relations. In particular, more expressive attribution methods are needed to express second-order task interactions. We provide the proof of this result in Appendix A.2.

Next, we experiment with a binary classification to empirically validate the connection, where the first-order approximation holds. We compare the estimates from both the linear task model and influence functions against the ground-truth leave-one-out (LOO) retraining scores. In Figure 1, we find that both methods produced estimates that aligned closely with the ground truth: the Linear Task Model and Influence Functions achieved high Pearson correlations of 0.98 and 0.97, respectively, with the LOO scores. The estimates from the two methods are also highly correlated with each other, with a Pearson correlation of 0.96. This demonstrates that when the performance landscape is nearly linear, both the empirical and analytical approaches indeed converge to the same underlying attribution scores. A detailed description of the experimental setup is available in Appendix B.1.

### 3.2 Learning kernel surrogate models

To build intuition for why a nonlinear data model is necessary, we present a numerical example in Figure 2. We use a binary classification task with a two-layer MLP as the base classifier. The final goal for the surrogate model is to predict the MLP output on a fixed test sample, given the subset of training data it was trained on. We specifically analyze the effect of different subsets of training data sampled from near the MLP decision boundary. The detailed setting is in Appendix B.1.

The influence of a training subset exhibits strong non-linearity that cannot be decomposed into individual sample contributions. When specific samples combine in a training subset, they produce changes in the MLP learned function, causing transitions in test point predictions. The linear surrogate model does not capture these interactions. By contrast, the RBF kernel surrogate model captures these dependencies by modeling the joint influence of sample combinations. This demonstrates that non-linear surrogate models are necessary for capturing nonlinear interactions.

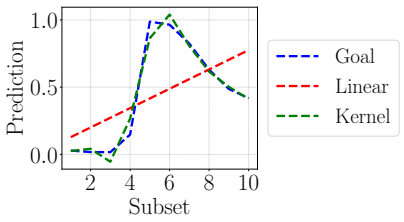



Figure 2: Illustrate linear vs. kernel models on a decision boundary.

**Estimating surrogate models using kernel ridge regression.** To address the limitation of linear surrogates, we propose kernel surrogate models, replacing linear mapping with a nonlinear function learned via kernel ridge regression. This enables modeling complex, non-additive task interactions.

We learn a kernel surrogate function $g_\theta : \{0,1\}^K \to \mathbb{R}$ within a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ by minimizing the following objective:

$$\min_{g_\theta \in \mathcal{H}_k} \sum_{i=1}^{m} \left(F(\mathbf{s}^{(i)}) - g_\theta(\mathbf{s}^{(i)})\right)^2 + \lambda \|g_\theta\|_{\mathcal{K}}^2, \tag{7}$$

Table 1: We investigate the surrogate model performance with different kernels.

| Residual error | Linear | RBF |
|---|---|---|
| CIFAR-10 | $4.4_{\pm 0.9}$ | $1.0_{\pm 0.0}$ |
| Modular arithmetic | $4.6_{\pm 1.3}$ | $1.5_{\pm 0.4}$ |
| In-context learning | $0.8_{\pm 0.2}$ | $0.4_{\pm 0.1}$ |
| Multi-objective RL | $0.2_{\pm 0.1}$ | $0.1_{\pm 0.1}$ |

Table 2: Relative approximation error of $\epsilon_W(x)$, tested on four different tasks.

| | Relative error |
|---|---|
| CIFAR-10 | $1.02_{\pm 0.69}\%$ |
| Modular arithmetic | $2.40_{\pm 2.17}\%$ |
| In-context learning | $0.51_{\pm 0.04}\%$ |
| Multi-objective RL | $0.43_{\pm 0.73}\%$ |

where $\lambda > 0$ is a regularization parameter and $\mathcal{K}$ is an $m \times m$ kernel matrix with entries $\mathcal{K}_{i,j} = k(\mathbf{s}^{(i)}, \mathbf{s}^{(j)})$. Since input vectors $\mathbf{s}$ are binary, we use the Radial Basis Function (RBF) kernel:

$$k(\mathbf{s}^{(a)}, \mathbf{s}^{(b)}) = \exp\big(-\gamma \|\mathbf{s}^{(a)} - \mathbf{s}^{(b)}\|^2\big). \tag{8}$$

By the representer theorem (Schölkopf et al., 2001), the minimizer takes the form:

$$g_\theta(\mathbf{s}) = \sum_{i=1}^{m} \theta_i k\big(\mathbf{s}^{(i)}, \mathbf{s}\big).$$

The coefficient vector $\theta = [\theta_1, \ldots, \theta_m]^\top$ is computed as: $\theta = (\mathcal{K} + \lambda \mathrm{Id})^{-1}\mathbf{F}$, where $\mathbf{F} = [F(\mathbf{s}^{(1)}), \ldots, F(\mathbf{s}^{(m)})]^\top$ is the vector of observed outcomes and Id denotes the identity matrix.

This kernel provides flexibility for capturing complex relationships while maintaining computational tractability. Hyperparameters $\lambda$ and $\gamma$ are selected via cross-validation. Unlike linear models that assign fixed coefficients to individual tasks, the kernel task model learns a global non-linear function that predicts performance for any task combination, thereby capturing intricate inter-dependencies.

We empirically validated this choice by comparing the RBF kernel with linear and polynomial kernels with degrees $\{1, 2, 3\}$ in Table 1. We use ResNet-9 network classifiers trained on the CIFAR-10 dataset (Krizhevsky & Hinton, 2009) as the base model. The detailed description is in Appendix B.2. We find that the RBF kernel achieves a much lower residual error compared to the linear surrogate model. We discuss more kernels in Appendix B.2.

**Remark 3.3.** *A major property of RBF kernels (Poggio & Girosi, 2002) is its universal expressivity over any distance functions on the space $\{0,1\}^K$. Moreover, the RBF kernel encodes an inductive bias that matches the geometry of the subset space $\{0,1\}^K$ (See also, the use of heat kernels for community detection (Kloster & Gleich, 2014)). We formalize this in Proposition A.3, Appendix A.5.*

### 3.3 Efficient estimation via projected gradients

One downside of kernel surrogate models is that they involve training $m$ separate models to obtain $F(\mathbf{s}^{(i)})$, one on each $\mathbf{s}^{(i)}$. Rather than pursuing this repeated training, we introduce an efficient algorithm to approximate the model outcomes instead. Consider a first-order Taylor expansion of model outputs around the initialization $W_0$ as follows:

$$f_W(x) = f_{W_0}(x) + \langle \nabla f_{W_0}(x), W - W_0 \rangle + \epsilon_W(x), \tag{9}$$

where $\epsilon_W(x)$ is the first-order Taylor expansion error of $f_{W_0}$ on input $x$. The intuition around this estimation algorithm can be related to how neural networks learn features through the gradients (Malladi et al., 2023; Radhakrishnan et al., 2024). Empirically, we also evaluate $\epsilon_W(x)$ relative to $f_W(x)$ in Table 2, which is with 2% across various test environments, and report further evaluation of this approximation on LLMs with up to 34B parameters in Table 5, Appendix A.4..

Next, we substitute approximation (9) of $f_W(x)$ into the empirical loss to estimate the optimal fine-tuning parameter perturbation on top of the initialization $W_0$ for every $\mathbf{s}^{(i)}$. Specifically, we formulate an approximate objective on each subset $\mathbf{s}^{(i)}$ as:

$$Z^\star_{\mathbf{s}^{(i)}} = \arg\min_{Z \in \mathbb{R}^d} \sum_{k=1}^{K} \frac{s_k^{(i)}}{\sum_{i=1}^{K} s_k^{(i)}} \sum_{(x,y)\in T_k} \ell_k\left(f_{W_0}(x) + \langle \nabla_W f_{W_0}(x), Z \rangle, y\right),$$

assuming the expansion error $\epsilon_W(x)$ is negligibly small. This corresponds to a (regularized) multinomial logistic regression problem, with (projected) gradients serving as features.

**Algorithm 1** Estimating Kernel Surrogate Models (KERNELSM)

**Input:** $K$ training tasks $T_1, \ldots, T_K$, pretrained model $f_{W_0}$, test dataset and metric $F(\cdot)$
**Requires:** Subset sampling distribution $\mathcal{D}$, number of subsets $m$, $\ell_2$-regularization parameter $\lambda$, kernel function $k(\cdot, \cdot)$
**Output:** An estimated kernel surrogate model (KSM)

/* *Generate random subsets and estimate test performance* */
$D_s \leftarrow \emptyset$: Initialize surrogate dataset
$\{f_{W_0}(z), \nabla f_{W_0}(z)\} \leftarrow$ Compute logits and gradients for all relevant samples $z$
**for** $i = 1, \ldots, m$ **do**
  $\mathbf{s}^{(i)} \sim \mathcal{D}$
  $\widehat{F}(\mathbf{s}^{(i)}) \leftarrow \texttt{GradEx}(\mathbf{s}^{(i)}, \{f_{W_0}(z), \nabla f_{W_0}(z)\}, F)$: Apply Algorithm 2
  $D_s \leftarrow D_s \cup \{(\mathbf{s}^{(i)}, \widehat{F}(\mathbf{s}^{(i)}))\}$
**end for**
/* *Learn a kernel surrogate model using kernel ridge regression* */
$\mathcal{K}_{i,j} \leftarrow k(\mathbf{s}^{(i)}, \mathbf{s}^{(j)})$: Construct the $m \times m$ kernel matrix $\mathcal{K}$
$\mathbf{y} \leftarrow [\widehat{F}(\mathbf{s}^{(1)}), \ldots, \widehat{F}(\mathbf{s}^{(m)})]^\top$: Compute the outcome vector
$\theta \leftarrow (\mathcal{K} + \lambda \cdot \mathrm{Id}_m)^{-1}\mathbf{y}$: KRR coefficients
**return** $g_\theta(\mathbf{s}) = \sum_{i=1}^m \theta_i k(\mathbf{s}^{(i)}, \mathbf{s})$

Based on this estimation algorithm, if we want to minimize the approximate loss on all the subsets $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(m)}$, we only need to compute the functional values and the gradients at the initialization $W_0$ (such as a pretrained LLM) once, including $f_{W_0}(x)$ and $\nabla f_{W_0}(x)$ for all $x$ in the training dataset. *This eliminates the need for repeated training and dramatically speeds up the estimation procedure, and is practical even for very large $m$ since the estimation procedure can be executed entirely on CPUs.*

Finally, we estimate the model output on a test sample $x, y$ using the same linear approximation:

$$\hat{f}(x) = f_{W_0}(x) + \langle \nabla f_{W_0}(x), Z^\star_{\mathbf{s}^{(i)}} \rangle.$$

The test performance of each $\mathbf{s}^{(i)}$ is then evaluated as the loss $\ell(\hat{f}(x), y)$ on the test dataset. Another challenge involved in this procedure is that the gradient vectors can have a large number of features. We address this problem by applying Gaussian random convolutions to project the gradient vectors into much lower dimensions. Then, solving the regression problem in the reduced space only takes several seconds, which can be run quickly on many subsets $m$. Due to space limit, we summarize the detailed procedure in Algorithm 2, Appendix A.3. Taken together, we summarize the overall procedure in Algorithm 1.

**Performance guarantees.** Assuming that the first-order approximation is small, we can bound the accuracy of the above estimation procedure relative to the true test performance using Johnson-Lindenstrauss lemma. For details, see the result statement and its proof following Proposition A.4.

## 4 EXPERIMENTS

Our experiments investigate three key questions: (1) How does KERNELSM compare to existing attribution techniques on complex tasks such as modular arithmetic reasoning, in-context learning, and multi-objective reinforcement learning? (2) Can the attributions generated by KERNELSM be effectively leveraged for downstream task selection? (3) How does a complex loss landscape impact the performance of KERNELSM compared to methods relying on linear surrogate models? We evaluate our approach across three settings, including arithmetic reasoning, in-context learning, and multi-objective reinforcement learning. We find that KERNELSM can improve over existing influence estimation methods by $25\%$. When applied to downstream task selection, KERNELSM can improve the performance of baselines by $41\%$. Various robustness checks and ablation studies validate the consistency of KERNELSM under different optimizers and sample sizes.

### 4.1 EXPERIMENT SETUP

**Datasets and models.** Our primary controlled environment is an arithmetic reasoning task, where we train a small decoder-only transformer. Consider learning a transformer model to perform modular arithmetic operations over two numbers, in the form of $a \circ b \pmod p = c$, where $\circ$ is an arithmetic operation and $p$ is a prime. Inputs are composed of four tokens: $a$, $\circ$, $b$, and $=$, with $a$ and $b$ generated between 0 and $p-1$. The label is the result of the arithmetic operation $c$. Specifically, we use the addition task $a + b \pmod p = c$ and the quadratic task $a^2 + b^2 + ab \pmod p = c$, where we set $p = 97$.

For more complex scenarios, we evaluate a Qwen3-8B model on in-context classification tasks. We construct prompts with $k = 4$ in-context examples followed by a query. For attribution computation, we treat each example's input embedding as the sample input, and the query's cross-entropy loss as the model score $F(\mathbf{s})$. We evaluate on two datasets, including SST-2 and coin flip.

We also evaluate attribution in multi-objective reinforcement learning using the Meta-World MT10 benchmark. It contains ten different manipulation tasks. We adopt the Soft Actor-Critic (SAC) algorithm as our training protocol. Each task is treated as a sample for attribution analysis, and we evaluate the model score by the average rewards of all tasks.

**Baselines.** We compare KERNELSM with existing data attribution methods. Influence functions (Koh & Liang, 2017) employ implicit differentiation, computing the inverse Hessian via LISSA. TracIn (Pruthi et al., 2020) traces prediction changes throughout training by computing gradient products between training and test examples. Linear surrogate models (Ilyas et al., 2022) train linear models that predict test behavior from binary indicators of training sample inclusion across multiple model retrainings. Trak (Park et al., 2023) applies random projection and the one-step Newton method to approximate the datamodel results without retraining. SOURCE (Bae et al., 2024) approximates unrolled differentiation by segmenting training into a few stationary phases and using gradient/Hessian statistics from a handful of checkpoints to estimate counterfactual parameter changes. Bayesian influence functions (BIF) (Kreer et al., 2025) extend classical influence functions by expressing influence as a covariance under a localized Bayesian posterior and estimating it via batched SGLD sampling.

**Evaluation.** We evaluate the data attribution methods by comparing them to the linear datamodeling score (LDS) (Ilyas et al., 2022). We compute LDS in the following steps: (1) Sample $m$ fixed subsets of the training set, which are represented by $\mathbf{s}^{(i)}, \ldots, \mathbf{s}^{(m)} \in \{0,1\}^K$. We sample $\mathbf{s}$ from a Bernoulli distribution with probability $p$. (2) For each $\mathbf{s}^{(i)}$, we train a model $F(\mathbf{s}^{(i)})$, and use a task attribution method to obtain an estimation $\hat{F}(\mathbf{s}^{(i)})$. (3) We compute the LDS from the Spearman correlation $\rho$ between the real model output and the task attribution estimation.

### 4.2 EXPERIMENTAL RESULTS

**Attribution results.** Our results in Table 3 demonstrate that by moving beyond the linear assumptions of prior work, KERNELSM achieves a more accurate estimation of task attribution. The advantage is most pronounced in tasks with complex, non-linear structures. For example, in modular arithmetic reasoning, KERNELSM improves the LDS by over $42\%$ on average compared to linear surrogate models. Similarly, on the reasoning Coin Flip ICL task, where the language model exhibits intricate prompt-following behavior that falters linear approximations, our method's ability to capture higher-order interactions yields a $25\%$ relative improvement in LDS. In multi-objective reinforcement learning, where the task's more transparent structure allows the linear baseline to perform well already (LDS of $0.76$), KERNELSM still shows an improvement to $0.80$.

We further evaluate whether KERNELSM scales to scenarios involving a large number of tasks. In our in-context learning experiments, we use $500$ prompt samples to simulate a task-scalable setting. The resulting LDS values remain comparable to the initial results tested on $50$ tasks: $0.31$ on SST-2 and $0.53$ on Coin-Flip. These results indicate that the performance of KERNELSM does not degrade as $K$ increases, demonstrating its robustness in scaling to settings with large $K$.

**Task selection results.** Next, we show that the accurate attributions from KERNELSM are applicable to downstream optimization tasks. We apply task attribution methods to find optimal groupings for downstream performance. For prompt selection in ICL, we select the top-4 examples with the

Table 3: Summary of comparison results on task attribution. We report the mean and standard deviation from five independent runs.

| Methods | Modular arithmetic reasoning | | In-context learning | | Multitask RL |
|---|---|---|---|---|---|
| | Addition | Quadratic | SST-2 | Coin flip | Metaworld |
| Influence functions | $0.03_{\pm 0.01}$ | $0.01_{\pm 0.01}$ | $0.16_{\pm 0.05}$ | $0.05_{\pm 0.10}$ | $0.71_{\pm 0.11}$ |
| TracIn | $0.17_{\pm 0.03}$ | $0.32_{\pm 0.09}$ | $0.21_{\pm 0.05}$ | $0.32_{\pm 0.09}$ | $0.45_{\pm 0.15}$ |
| TRAK | $0.14_{\pm 0.01}$ | $0.28_{\pm 0.06}$ | $0.11_{\pm 0.08}$ | $0.23_{\pm 0.12}$ | $0.42_{\pm 0.19}$ |
| SOURCE | $0.22_{\pm 0.05}$ | $0.28_{\pm 0.06}$ | / | / | $0.27_{\pm 0.19}$ |
| BIF | $0.18_{\pm 0.16}$ | $0.42_{\pm 0.07}$ | $0.14_{\pm 0.02}$ | $0.28_{\pm 0.06}$ | $0.22_{\pm 0.24}$ |
| Linear surrogate models | $0.18_{\pm 0.02}$ | $0.44_{\pm 0.09}$ | $0.33_{\pm 0.05}$ | $0.43_{\pm 0.05}$ | $0.76_{\pm 0.04}$ |
| KERNELSM | $\mathbf{0.30}_{\pm 0.12}$ | $\mathbf{0.52}_{\pm 0.08}$ | $\mathbf{0.37}_{\pm 0.02}$ | $\mathbf{0.54}_{\pm 0.01}$ | $\mathbf{0.80}_{\pm 0.04}$ |

Table 4: We evaluate KERNELSM on the downstream task selection. We measure performance by the loss in ICL tasks and the target rewards in multi-objective RL tasks. We measure the running time in minutes.

| Loss (↓) | SST-2 | Coin flip | Time (↓) | Rewards (↑) | MT 10 | Time (↓) |
|---|---|---|---|---|---|---|
| Influence func. | $0.23_{\pm 0.21}$ | $1.62_{\pm 1.33}$ | $17_{\pm 1}$ | Influence func. | $16.3_{\pm 1.4}$ | $2_{\pm 1}$ |
| TracIn | $0.25_{\pm 0.40}$ | $0.54_{\pm 0.68}$ | $3_{\pm 1}$ | TracIn | $16.0_{\pm 2.3}$ | $3_{\pm 1}$ |
| Trak | $0.32_{\pm 0.48}$ | $0.72_{\pm 0.87}$ | $2_{\pm 1}$ | Trak | $15.8_{\pm 4.5}$ | $1_{\pm 1}$ |
| Lin. surrogate | $0.20_{\pm 0.11}$ | $0.05_{\pm 0.03}$ | $2_{\pm 1}$ | Lin. surrogate | $13.1_{\pm 4.3}$ | $1_{\pm 1}$ |
| KERNELSM | $0.16_{\pm 0.08}$ | $0.02_{\pm 0.03}$ | $2_{\pm 1}$ | KERNELSM | $18.8_{\pm 5.6}$ | $1_{\pm 1}$ |

highest attribution score. For synergistic task selection in RL, we co-train each target task with the top-3 tasks identified by our method to encourage positive transfer.

To derive an individual attribution for each item from the kernel surrogate model, we adopt a random ensemble method. We sample multiple subsets and use our kernel surrogate to obtain a performance prediction for each. The final attribution for any given item is then calculated as the average prediction score of all sampled subsets in which that item was included.

As shown in Table 4, KERNELSM achieves a $40\%$ lower loss in the prompt selection task in ICL, and improves the target rewards by $15\%$. These results demonstrate the benefit of using KERNELSM over linear surrogate models.

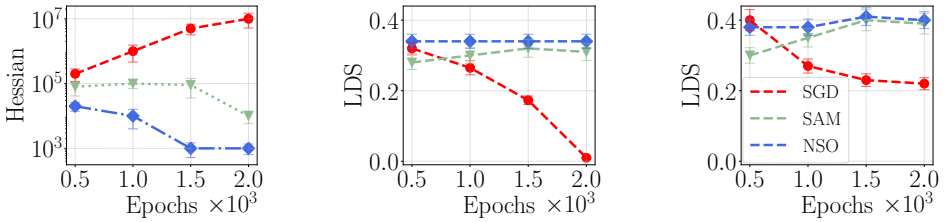
### 4.3 REGULARIZATION OF HESSIAN

Recall from equation (5) that the error of surrogate models is influenced by the second-order terms. Next, we compare the accuracy of linear task modeling by measuring the curvature of the loss Hessian $\nabla^2_W \hat{L}(\hat{W})$, by controlling it with different Hessian regularization. We report results on a modular quadratic task trained using a two-layer transformer. We compared three methods, including the standard SGD without explicit Hessian regularization, and two Hessian regularization methods: sharpness-aware minimization (SAM) (Foret et al., 2021), and noise stability optimization (NSO) (Zhang et al., 2024). We measure the Hessian trace on the global model, and the LDS and residual error of the linear surrogate model are measured on subsets with $\alpha = 0.9$.

In SGD, the Hessian trace increases continuously throughout the training process. This worsens the linear surrogate model's performance, characterized by a decrease in LDS. Eventually, the LDS dropped to $0.01$. By contrast, both SAM and NSO constrain the Hessian trace. This regularization results in improved model fit, whose LDS remained relatively consistent throughout training.

## 5 RELATED WORK

Our work is related to the literature on training data attribution methods. Data attribution methods have been extensively studied to quantify the influence of individual training samples on model behavior (Koh & Liang, 2017; Brown et al., 2021; Ilyas et al., 2022). In this work, we extend these concepts by measuring the contribution of groups of samples that form different training tasks.




(a) Hessian trace (b) Linear model (c) RBF Kernel

Figure 3: We investigate both linear and kernel surrogate models' fit under different Hessian regularization during model training. The linear surrogate model does not fit model outcomes when using SGD, and only works when its Hessian is regularized. By contrast, kernel surrogate models remain robust when trained with various optimizers and regularization.

The first line of approach is based on the influence function (Koh & Liang, 2017). These methods compute the influence function. The main challenge is that the influence function involves Hessian computation. Instead of computing the exact Hessian inverse, recent approaches use the Gauss-Newton Hessian approximation (Choe et al., 2024) or sparse gradient compression (Hu et al., 2025). Influence functions can also be applied to non-decomposable loss functions, such as contrastive loss and preference loss, allowing for data attribution across a broader class of models (Deng et al., 2025). The second line of approach studies the trajectory of model training, rather than just examining the final state of the trained model. These methods aim to trace the training dynamics and predict the counterfactual trajectory that would result from a different training set (Bae et al., 2024). Recent work calculates the exact influence of training data on a single, deterministic model instance by leveraging large-scale meta-gradient computation (Engstrom et al., 2025; Ilyas & Engstrom, 2025). The third line of research involves using a surrogate model to approximate the behavior of the original complex model and then computing the surrogate model. Recent work shows that the model output can be linearized in a certain local area (Li et al., 2024b). Under this assumption, one can rephrase the model output as a first-order Taylor expansion on the parameters. This approach provides an efficient estimation for the leave-one-out score (Li et al., 2023) and for datamodels (Park et al., 2023). Building on surrogate models, our work uses a kernel-based model to better capture the non-linear interactions between tasks. The use of kernel ridge regression, also known as Tikhonov regularization, can be related to earlier literature on kernel methods (Dicker et al., 2017), and it may be worth further examine the role of regularization in task attribution. The RBF kernel that we have used is also related to the heat kernel, which is highly effective for seed set expansion in community detection (Kloster & Gleich, 2014).

A limitation of this work is that our evaluation primarily focuses on prediction tasks. Extending our attribution framework to examine other aspects of language models such as planning abilities remains an open question, which is a fundamental aspect of human intelligence (Wang et al., 2024). In particular, this approach may explain the generalization gaps between SFT and RL (Wang et al., 2025); we leave a detailed exploration of this connection to future work.

## 6 Conclusion

In this paper, we study the problem of task attribution, which aims to quantify the influence of different training tasks on a model's final performance. We first unify the theories of existing influence functions and linear surrogate models, highlighting their limitations in handling non-linear interactions between tasks. To overcome this challenge, we propose a new kernel-based attribution method. This approach effectively captures complex task interactions and does not rely on specific assumptions about the training process. This makes it broadly applicable to emerging paradigms, including in-context learning and reinforcement learning. Furthermore, we introduce an efficient gradient approximation technique that avoids expensive retraining, ensuring our method is scalable. We validate our method's effectiveness with experiments across several distinct domains. The experiment results demonstrate that our approach provides more accurate attribution than existing methods in a diverse range of settings that are relevant to modern AI systems.

## ACKNOWLEDGMENT

Thanks to Chris Ré for insightful discussions about this work. We appreciate the anonymous referees for their constructive comments and suggestions. This work is supported in part by NSF award IIS-2412008. The views and opinions expressed in this paper are solely those of the authors and do not necessarily reflect those of the National Science Foundation.

## REFERENCES

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021.

Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, pp. 123–132, 2021.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Grosse, and Eric Xing. What is your data worth to gpt? llm-scale data valuation with influence functions. *arXiv preprint arXiv:2405.13954*, 2024.

Junwei Deng, Weijing Tang, and Jiaqi W. Ma. A versatile influence function for data attribution with non-decomposable loss. In *Forty-second International Conference on Machine Learning*, 2025.

Samuel Deng and Daniel Hsu. Multi-group learning for hierarchical groups. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 10440–10487, 2024.

Lee H Dicker, Dean P Foster, and Daniel Hsu. Kernel ridge vs. principal component regression: Minimax bounds and the qualification of regularization operators. *Electronic Journal of Statistics*, 2017.

Logan Engstrom, Andrew Ilyas, Benjamin Chen, Axel Feldmann, William Moses, and Aleksander Madry. Optimizing ml training with metagradient descent. *arXiv preprint arXiv:2503.13751*, 2025.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Pingbang Hu, Joseph Melkonian, Weijing Tang, Han Zhao, and Jiaqi W Ma. Grass: Scalable influence function with sparse gradient compression. *arXiv preprint arXiv:2505.18976*, 2025.

Andrew Ilyas and Logan Engstrom. Magic: Near-optimal data attribution for deep learning. *arXiv preprint arXiv:2504.16430*, 2025.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. In *International conference on machine learning*, 2022.

William B Johnson. Extensions of lipshitz mapping into hilbert space. In *Conference modern analysis and probability, 1984*, pp. 189–206, 1984.

Kyle Kloster and David F Gleich. Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1386–1395, 2014.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Philipp Alexander Kreer, Wilson Wu, Maxwell Adam, Zach Furman, and Jesse Hoogland. Bayesian influence functions for scalable data attribution. In *High-dimensional Learning Dynamics*, 2025.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

Dongyue Li, Huy Nguyen, and Hongyang R. Zhang. Identification of negative transfers in multitask learning using surrogate models. *Transactions on Machine Learning Research*, 2023.

Dongyue Li, Aneesh Sharma, and Hongyang R Zhang. Scalable multitask learning using gradient-based estimation of task affinity. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1542–1553, 2024a.

Dongyue Li, Ziniu Zhang, Lu Wang, and Hongyang R. Zhang. Scalable fine-tuning from multiple data sources: A first-order approximation approach. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 5608–5623, 2024b.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A kernel-based view of language model fine-tuning. In *International Conference on Machine Learning*, pp. 23610–23641. PMLR, 2023.

Charles A Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. In *Approximation theory and spline functions*, pp. 143–145. Springer, 1984.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, 2022.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning*, pp. 27074–27113. PMLR, 2023.

Tomaso Poggio and Federico Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 2002.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Adityanarayanan Radhakrishnan, Daniel Beaglehole, Parthe Pandit, and Mikhail Belkin. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pp. 416–426. Springer, 2001.

Siwei Wang, Yifei Shen, Shi Feng, Haoran Sun, Shang-Hua Teng, and Wei Chen. Alpine: Unveiling the planning capability of autoregressive learning in language models. *Advances in neural information processing systems*, 37:119662–119688, 2024.

Siwei Wang, Yifei Shen, Haoran Sun, Shi Feng, Shang-Hua Teng, Li Dong, Yaru Hao, and Wei Chen. Benefits and pitfalls of reinforcement learning for language model planning: a theoretical perspective. *arXiv preprint arXiv:2509.22613*, 2025.

Chenwei Wu, Holden Lee, and Rong Ge. Connecting pre-trained language model and downstream task via properties of representation. *Advances in Neural Information Processing Systems*, 36: 47216–47238, 2023.

Sen Wu, Hongyang R. Zhang, and Christopher Ré. Understanding and improving information transfer in multi-task learning. In *International Conference on Learning Representations*, 2020.

Fan Yang, Hongyang R. Zhang, Sen Wu, Christopher Ré, and Weijie Su. Precise high-dimensional asymptotics for quantifying heterogeneous transfers. *Journal of Machine Learning Research*, 26 (113):1–88, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Hongyang R. Zhang, Dongyue Li, and Haotian Ju. Noise stability optimization for finding flat minima: A hessian-based regularization approach. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Zhenshuo Zhang, Minxuan Duan, Youran Ye, and Hongyang R. Zhang. Scalable multi-objective and meta reinforcement learning via gradient estimation. In *AAAI*, 2026.

Ziniu Zhang, Zhenshuo Zhang, Dongyue Li, Lu Wang, Jennifer Dy, and Hongyang R. Zhang. Linear-time demonstration selection for in-context learning via gradient estimation. *Empirical Methods in Natural Language Processing*, 2025.

# A COMPLETE PROOFS

**Derivation of influence functions.** Let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a dataset including $n$ samples drawn independently from an unknown data distribution. Let $f_W$ denote a model with parameters $W \in \mathbb{R}^d$. Let $\hat{L}(f_W)$ denote the empirical loss of the model $f_W$ on $S$, averaged over the $n$ training data samples. The influence function (Koh & Liang, 2017), which is defined by how much a trained model changes after adding or removing one sample $z = (x, y)$, is given by

$$[\nabla^2 \hat{L}(f_W)]^{-1} \nabla \ell(f_W(x), y).$$

To derive this result, let the perturbed parameter after adding input $z$ with step size $\epsilon$ be given by:

$$\widehat{W}_{\epsilon,z} = \arg\min_W \left( \hat{L}(f_W) + \epsilon \cdot \ell(f_W(x), y) \right). \tag{10}$$

Define the parameter change $\Delta_\epsilon = \widehat{W}_{\epsilon,z} - \widehat{W}$ and note that, since $\widehat{W}$ does not depend on $\epsilon$, we can thus write:

$$\frac{d\widehat{W}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}.$$

Based on first-order optimality conditions, from equation (10), we get:

$$\nabla \hat{L}(f_{\widehat{W}_{\epsilon,z}}) + \epsilon \cdot \nabla \ell(f_{\widehat{W}_{\epsilon,z}}(x), y) = 0.$$

Since $\widehat{W}_{\epsilon,z} \to \widehat{W}$ as $\epsilon \to 0$, we perform Taylor expansion centered at $\widehat{W}$ up to first-order as:

$$\left( \nabla \hat{L}(f_{\widehat{W}}) + \epsilon \cdot \nabla \ell(f_{\widehat{W}_{\epsilon,z}}(x), y) \right) + \left( \nabla^2 \hat{L}(f_{\widehat{W}}) + \epsilon \cdot \nabla^2 \ell(f_{\widehat{W}}(x), y) \right) \Delta_\epsilon \approx 0,$$

which is approximately zero after dropping the second-order term.

After setting $\epsilon$ to approaching zero, and solving for $\Delta_\epsilon$, we get:

$$\Delta_\epsilon = -[\nabla^2 \hat{L}(f_{\widehat{W}})]^{-1} \left( \nabla \hat{L}(f_{\widehat{W}}) + \epsilon \nabla \ell(f_{\widehat{W}}(x), y) \right) = -\epsilon [\nabla^2 \hat{L}(\widehat{W})]^{-1} \nabla \ell(f_{\widehat{W}}(x), y),$$

since $\nabla \hat{L}(f_{\widehat{W}}) = 0$. Through the chain rule, the influence on any differentiable function $F(\mathbf{s})$ becomes:

$$\mathcal{I}_z(\mathbf{s}) = [\nabla_W F(\mathbf{s})]^\top \left[ \nabla^2_W \hat{L}(f_W) \right]^{-1} \nabla \ell(f_W(x), y).$$

By aggregating over $k = 1, 2, \ldots, K$, we conclude the proof of equation (2).

**Notations.** Given a square matrix $X \in \mathbb{R}^{d \times d}$, we use $\operatorname{diag}[X] \in \mathbb{R}^d$ to denote the diagonal entries of $X$ organized as a vector. Let $\|X\|_2$ denote the largest singular value of $X$. Let $\|x\|$ denote the Euclidean norm of a vector $x \in \mathbb{R}^d$.

We follow the convention of big-O notations. We use $f(n) = O(g(n))$ to indicate that there exists a constant $c > 0$ so that for large enough $n$, $f(n) \leq c \cdot g(n)$. We also write $f(n) \lesssim g(n)$ to indicate that $f(n) = O(g(n))$.

## A.1 PROOF OF PROPOSITION 3.1

*Proof.* We denote by $\mu = \mathbb{E}[\mathbf{s}] = [p]_K$. First, we derive the population ordinary least squares (OLS) coefficients for the linear predictor

$$g(\mathbf{s}) = \alpha + \sum_{j=1}^{K} \beta_j \mathbf{s}_j.$$

Let $\alpha^\star$ and $\beta^\star = (\beta_1^\star, \ldots, \beta_K^\star)$ denote the minimizers of the population squared loss

$$\alpha^\star, \beta^\star \leftarrow \arg\min_{\alpha,\beta} \mathop{\mathbb{E}}_{\mathbf{s} \sim \mathcal{D}} \left[ \left( F(\mathbf{s}) - \alpha - \sum_{j=1}^{K} \beta_j \mathbf{s}_j \right)^2 \right].$$

Taking the derivative with respect to $\alpha$ and setting it to zero gives

$$\mathbb{E}\Big[F(\mathbf{s}) - \alpha^\star - \sum_{j=1}^{K} \beta_j^\star \mathbf{s}_j\Big] = 0, \tag{11}$$

We differentiate with respect to each $\beta_i^\star$ and set the derivative to zero:

$$\mathbb{E}\Big[\mathbf{s}_i\big(F(\mathbf{s}) - \alpha^\star - \sum_{j=1}^{K} \beta_j^\star \mathbf{s}_j\big)\Big] = 0, \qquad i = 1, \ldots, K.$$

Expanding the expectation and using linearity yields

$$\mathbb{E}[\mathbf{s}_i F(\mathbf{s})] - \alpha^\star \mathbb{E}[\mathbf{s}_i] - \sum_{j=1}^{K} \beta_j^\star \, \mathbb{E}[\mathbf{s}_i \mathbf{s}_j] = 0. \tag{12}$$

From equation (11), we have

$$\alpha^\star = \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}}[F(\mathbf{s})] - \sum_{j=1}^{K} \beta_j^\star \mathop{\mathbb{E}}_{\mathbf{s}\sim\mathcal{D}}[\mathbf{s}_j]\,. \tag{13}$$

By multiplying both sides of equation (13) with $\mathbf{s}_i$ and taking expectation on $\mathbf{s}$ again, we get

$$\mathbb{E}[\mathbf{s}_i]\,\mathbb{E}[F(\mathbf{s})] = \sum_{j=1}^{K} \beta_j \,\mathbb{E}[\mathbf{s}_i]\,\mathbb{E}[\mathbf{s}_j]\,. \tag{14}$$

Subtracting and adding both sides from equation (14) into equation (12) leads to the following covariance form

$$\mathrm{Cov}[\mathbf{s}_i, F(\mathbf{s})] = \sum_{j=1}^{K} \beta_j^\star \,\mathrm{Cov}[\mathbf{s}_i, \mathbf{s}_j], \qquad i = 1, \ldots, K,$$

which are the population normal equations. Since subsets are independent, we have

$$\mathrm{Cov}[\mathbf{s}_i, \mathbf{s}_j] = 0 \quad \text{for } i \neq j,$$

so the covariance matrix of $\mathbf{s}$ is diagonal. Thus each normal equation decouples:

$$\mathrm{Cov}[\mathbf{s}_k, F(\mathbf{s})] = \beta_k^\star \,\mathrm{Var}[\mathbf{s}_k].$$

Solving for $\beta_k^\star$ gives the uni-variate OLS coefficient

$$\hat{\beta}_k^\star = \frac{\mathrm{Cov}[\mathbf{s}_k, F(\mathbf{s})]}{\mathrm{Var}[\mathbf{s}_k]}. \tag{15}$$

Next, we decompose the covariance using second-order Taylor expansion as:

$$\mathrm{Cov}[\mathbf{s}_k, F(\mathbf{s})] = \underbrace{\mathrm{Cov}[\mathbf{s}_k, F(\mathbf{s}^\star)]}_{A_1} + \underbrace{\mathrm{Cov}\big[\mathbf{s}_k, \mathbf{G}_\mathbf{s}^\top(\mathbf{s} - \mathbf{s}^\star)\big]}_{A_2} + \underbrace{\mathrm{Cov}\big[\mathbf{s}_k, \tfrac{1}{2}(\mathbf{s} - \mathbf{s}^\star)^\top \mathbf{H}_\mathbf{s}(\mathbf{s} - \mathbf{s}^\star)\big]}_{A_3} + R_3(\mathbf{s}_k),$$

where we denote the gradient and Hessian as $\mathbf{G}_\mathbf{s}, \mathbf{H}_\mathbf{s}$, respectively, and the expansion error term as $R_3(\mathbf{s}_k)$. Notice that $A_1 = 0$. Thus, $A_2$ is the first-order term, and $A_3$ is the second-order term.

Next, by the independence between different coordinates of $\mathbf{s}$, we have that

$$A_2 = g_k \mathrm{Var}[\mathbf{s}_k],$$

where $g_k = [\nabla_\mathbf{s} F(\mathbf{s})]_k$.

For $A_3$, write $\mathbf{s} - \mathbf{s}^\star = (\mathbf{s} - \mu) + (\mu - \mathbf{s}^\star)$ and expand the Hessian product as:

$$Q = \tfrac{1}{2}\Big((\mathbf{s} - \mu)^\top \mathbf{H}_\mathbf{s}(\mathbf{s} - \mu) + 2(\mu - \mathbf{s}^\star)^\top \mathbf{H}_\mathbf{s}(\mathbf{s} - \mu) + (\mu - \mathbf{s}^\star)^\top \mathbf{H}_\mathbf{s}(\mu - \mathbf{s}^\star)\Big).$$

For the second term above,

$$\mathrm{Cov}\big[\mathbf{s}_k, (\mu - \mathbf{s}^\star)^\top \mathbf{H}_\mathbf{s}(\mathbf{s} - \mu)\big] = \mathrm{Var}[\mathbf{s}_k] \sum_{j=1}^{K} \mathbf{H}_\mathbf{s}(k,j)(\mu_j - \mathbf{s}_j^\star),$$

using symmetry of $\mathbf{H}_\mathbf{s}$, where $\mathbf{H}_\mathbf{s}(k,j)$ denotes the $(k,j)$-th entry of $\mathbf{H}_\mathbf{s}$. For the first term,

$$(\mathbf{s} - \mu)^\top \mathbf{H}_\mathbf{s}(\mathbf{s} - \mu) = \sum_{i=1}^{K} \mathbf{H}_\mathbf{s}(i,i)(\mathbf{s}_i - \mu_i)^2 + 2 \sum_{1 \le i < j \le K} \mathbf{H}_\mathbf{s}(i,j)(\mathbf{s}_i - \mu_i)(\mathbf{s}_j - \mu_j).$$

All cross terms with $i \neq j$ vanish by independence and zero mean of $\mathbf{s}_j - \mu_j$, leaving

$$\tfrac{1}{2}\mathrm{Cov}\big[\mathbf{s}_k, (\mathbf{s} - \mu)^\top \mathbf{H}_\mathbf{s}(\mathbf{s} - \mu)\big] = \tfrac{1}{2}\mathbf{H}_\mathbf{s}(k,k)\mathrm{Cov}\big[\mathbf{s}_k, (\mathbf{s}_k - \mu_k)^2\big].$$

With $X_k = \mathbf{s}_k - \mu_k$, $\mathrm{Cov}[\mathbf{s}_k, X_k^2] = \mathbb{E}[X_k^3]$, and for Bernoulli$(p)$,

$$\mathbb{E}[X_k^3] = p(1-p)(1-2p).$$

Hence

$$\mathrm{Cov}[\mathbf{s}_k, Q] = \mathrm{Var}[\mathbf{s}_k] \sum_{j=1}^{K} \mathbf{H}_\mathbf{s}(k,j)(\mu_j - \mathbf{s}_j^\star) + \tfrac{1}{2}\mathbf{H}_\mathbf{s}(k,k)p(1-p)(1-2p), \tag{16}$$

which yields

$$\left| \beta_k^\star - g_k - \sum_{j=1}^{K} \mathbf{H}_\mathbf{s}(k,j)(\mu_j - \mathbf{s}_j^\star) - \tfrac{1}{2}\mathbf{H}_\mathbf{s}(k,k)(1-2p) \right| \le R_3(\mathbf{s}_k)/\mathrm{Var}[\mathbf{s}_k]. \tag{17}$$

By applying equation (17) over $k = 1, 2, \ldots, K$, we have shown that

$$\left\| \beta^\star - \nabla_\mathbf{s} F(\mathbf{s}^\star) - (p-1)\mathbf{H}_\mathbf{s}[\mathbf{1}]_K - \frac{1-2p}{2}\,\mathrm{diag}\,[\mathbf{H}_\mathbf{s}] \right\| \le \sum_{k=1}^{K} R_3(\mathbf{s}_k)/(p(1-p)), \tag{18}$$

where $\mathrm{Var}[\mathbf{s}_k] = p(1-p)$ since $\mathbf{s}_k$ is a Bernoulli random variable with probability $p$. Assuming the third-order derivatives of $F(\mathbf{s})$ are bounded by some small enough $c_3$, then, we have that

$$\sum_{k=1}^{K} R_3(\mathbf{s}_k) = O\left(c_3\, \mathbb{E}\left[\|s - s^\star\|^3\right]\right) \le O\left(8c_3 K^{3/2}(1-p)\right). \tag{19}$$

By applying equation (19) to equation (18), we obtain an error term as $O(c_3 K^{3/2} p^{-1})$.

The last step is to bound the minimizer between the empirical loss $\hat{R}(\alpha, \beta)$ and the population loss $R(\alpha, \beta)$. Recall that $\hat{\beta}$ is the empirical risk minimizer on $m$ samples. Since we are working a noiseless setting, we have that

$$\left\| \hat{\beta} - \beta^\star \right\| \le C \left\| \beta^\star \right\| \kappa(\Sigma) \sqrt{\frac{K + \log(\delta^{-1})}{m}}, \tag{20}$$

for some fixed constant $C > 0$. This can be shown using standard OLS analysis and concentration bounds on the sample covariance matrix. In the case of Bernoulli sampling, we have that the population covariance matrix is $p(1-p)\,\mathrm{Id}$, thus $\kappa(\Sigma) = 1$. Thus, by combining equation (20) with the Taylor expansion errors above, we have concluded the proof of equation (5). □

**Corollary A.1.** *In the setting of Proposition 3.1, the intercept and the residual error of linear surrogate models are given by*

$$\hat{\alpha} = \mathbb{E}[F(\mathbf{s})] - \hat{\beta}^\top \mu,$$

$$\min_{\alpha,\beta} \mathbb{E}\big[(F(\mathbf{s}) - \alpha - \beta^\top \mathbf{s})^2\big] = \mathrm{Var}[Q] - \sum_{k=1}^{K} \frac{\mathrm{Cov}[\mathbf{s}_k, Q]^2}{\mathrm{Var}[\mathbf{s}_k]},$$

*where $Q$ is defined in equation (16).*

*Proof.* First, the intercept term follows from equation (13).

Next, consider the residual mean-squared error term. Let $X = \mathbf{s} - \mu$ (zero mean, independent coordinates). The optimal linear predictor removes the constant and all components in the span of $\{X_k\}$. Therefore

$$\mathcal{E} = \mathrm{Var}\Big[Q - \mathbb{E}[Q] - \sum_{k=1}^{K} b_k X_k\Big] = \mathrm{Var}[Q] - \sum_{k=1}^{K} \frac{\mathrm{Cov}[X_k, Q]^2}{\mathrm{Var}[X_k]}.$$

Because $\mathrm{Cov}[X_k, Q] = \mathrm{Cov}[\mathbf{s}_k, Q]$ and $\mathrm{Var}[X_k] = \mathrm{Var}[\mathbf{s}_k] = p(1-p)$, this gives

$$\mathcal{E} \;\equiv\; \min_{\alpha,\beta} \mathbb{E}\left[(F(\mathbf{s}) - \alpha - \beta^\top \mathbf{s})^2\right] \;\approx\; \mathrm{Var}[Q] - \sum_{k=1}^{K} \frac{\mathrm{Cov}[\mathbf{s}_k, Q]^2}{\mathrm{Var}[\mathbf{s}_k]}. \tag{21}$$

It remains to compute $\mathrm{Var}[Q]$. Since $Q = \frac{1}{2} X^\top \mathbf{H}_\mathbf{s} X$,

$$X^\top \mathbf{H}_\mathbf{s} X = \sum_{i=1}^{K} \mathbf{H}_\mathbf{s}(i,i) X_i^2 + 2 \sum_{1 \le i < j \le K} \mathbf{H}_\mathbf{s}(i,j) X_i X_j$$

$$\Rightarrow \mathrm{Var}[X^\top \mathbf{H}_\mathbf{s} X] = \sum_i \mathbf{H}_\mathbf{s}(i,i)^2 \mathrm{Var}[X_i^2] + 4 \sum_{1 \le i < j \le K} \mathbf{H}_\mathbf{s}(i,j)^2 \mathrm{Var}[X_i X_j],$$

by independence and mean zero, eliminating mixed covariances. Thus

$$\mathrm{Var}[Q] = \tfrac{1}{4} \mathrm{Var}[X^\top \mathbf{H}_\mathbf{s} X].$$

For Bernoulli$(p)$, $X_i \in \{1-p, -p\}$ with

$$\mathbb{E}[X_i^2] = p(1-p),$$
$$\mathbb{E}[X_i^4] = p(1-p)^4 + (1-p)p^4.$$

Thus,

$$\mathrm{Var}[X_i^2] = \mathbb{E}[X_i^4] - \mathbb{E}[X_i^2]^2 = p(1-p)(1-2p)^2,$$
$$\mathrm{Var}[X_i X_j] = \mathbb{E}[X_i^2]\mathbb{E}[X_j^2] = (p(1-p))^2 \; (i \neq j).$$

Substituting back to $Q$ gives

$$\mathrm{Var}[Q] = \tfrac{1}{4} \left[ \sum_{i=1}^{K} \mathbf{H}_\mathbf{s}(i,i)^2 p(1-p)(1-2p)^2 + 4 \sum_{1 \le i < j \le K} \mathbf{H}_\mathbf{s}(i,j)^2 \big(p(1-p)\big)^2 \right]. \tag{22}$$

Plugging equations (16) and (22) into equation (21) gives the detailed residual mean-squared error. □

### A.2 Proof of Corollary 3.2

Next, we build on the above proof to derive Corollary 3.2. We show that the influence function estimates the gradients of the performance function. Our goal is to show that the vector of influence functions $\vec{\mathcal{I}} = (\mathcal{I}_1, \ldots, \mathcal{I}_K)^\top$ is equal to the gradient vector $\nabla_\mathbf{s} F(\mathbf{s})$.

The optimal parameters $\hat{W}(\mathbf{s}) = \arg\min_W \hat{L}(f_W, \mathbf{s})$. The empirical risk minimizer corresponds to the uniform weight vector $\mathbf{s}^\star = [\mathbf{1}]_K$. Let $F(\mathbf{s})$ be any differentiable performance metric that depends on the optimal parameters, e.g., the loss on a test sample $(x, y)$, $F(\mathbf{s}) = \ell(f_{\widehat{W}(\mathbf{s})}(x), y)$. The gradient of this function is given by $\nabla_\mathbf{s} F(\mathbf{s}^\star)$.

*Proof.* Recall that the influence function uses an additive perturbation $\epsilon$ for a single task $i$. The perturbed objective is:

$$\tilde{L}(f_W, \epsilon) = \left( \frac{1}{K} \sum_{j=1}^{K} \ell(f_W(x_j), y_j) \right) + \frac{\epsilon}{K} \ell(f_W(x_i), y_i).$$

We can rewrite this expression by collecting terms for each sample:

$$\tilde{L}(f_W, \epsilon) = \frac{1}{K}\left((1+\epsilon)\ell(f_W(x_i), y_i) + \sum_{j \neq i} \ell(f_W(x_j), y_j)\right).$$

Thus, the additive perturbation is a specific instance that corresponds to a path in the $K$-dimensional weight space $\mathbf{s}$:

$$\mathbf{s}(\epsilon) = (1, 1, \ldots, \underbrace{1+\epsilon}_{i\text{-th position}}, \ldots, 1)^\top = \mathbf{s}^\star + \epsilon \cdot e_i.$$

where $e_i$ refers to the $i$-th standard basis vector.

The influence function for task $i$, $\mathcal{I}_i(\mathbf{s})$, is defined as the derivative of $F$ with respect to $\epsilon$, evaluated at $\epsilon = 0$. We can compute this derivative using the multivariate chain rule on $F(\mathbf{s})$ as:

$$\mathcal{I}_i(\mathbf{s}) = \left.\frac{dF(\mathbf{s}(\epsilon))}{d\epsilon}\right|_{\epsilon=0}.$$

The chain rule states:

$$\frac{dF(\mathbf{s}(\epsilon))}{d\epsilon} = \left[\nabla_{\mathbf{s}} F(\mathbf{s}(\epsilon))\right]^\top \cdot \frac{d\mathbf{s}(\epsilon)}{d\epsilon}. \tag{23}$$

By definition, $\mathbf{s}(\epsilon) = \mathbf{s}^\star + \epsilon \cdot e_i$, thus

$$\frac{d\mathbf{s}(\epsilon)}{d\epsilon} = e_i.$$

Substituting this back to equation (23), we get:

$$\frac{dF(\mathbf{s}(\epsilon))}{d\epsilon} = \left[\nabla_{\mathbf{s}} F(\mathbf{s}(\epsilon))\right]^\top e_i = \frac{\partial F(\mathbf{s}(\epsilon))}{\partial \mathbf{s}_i}.$$

Thus, the derivative of $F(\mathbf{s}(\epsilon))$ with respect to $\epsilon$ is exactly the partial derivative with respect to the weight $\mathbf{s}_i$. Finally, consider $\epsilon = 0$. We have $\mathbf{s}(0) = \mathbf{s}^\star$. Therefore:

$$\mathcal{I}_i(\mathbf{s}) = \left.\frac{\partial F\left(\mathbf{s}(\epsilon)\right)}{\partial \epsilon}\right|_{\epsilon=0} = \left.\frac{\partial F(\mathbf{s})}{\partial \mathbf{s}_i}\right|_{\mathbf{s}=\mathbf{s}^\star}.$$

Since this holds for all $i = 1, \ldots, K$, the vector of influence functions is identical to the gradient vector of the performance function:

$$\vec{\mathcal{I}} = [\mathcal{I}_1(\mathbf{s}^\star), \ldots, \mathcal{I}_K(\mathbf{s}^\star)]^\top = \nabla_{\mathbf{s}} F(\mathbf{s}^\star).$$

Next, recall that $\|\mathbf{H}_{\mathbf{s}}\|_2 \leq c_2$ by assumption. As a result, the sampling bias and variance terms in equation (5) are at most:

$$c_2\sqrt{K} + c_2/2.$$

In summary, we have shown that $\left\|\beta^\star - \vec{\mathcal{I}}\right\| \leq c_2\sqrt{K} + c_2/2$. Together with the concentration bound between $\hat{\beta}$ and $\beta^\star$ from equation (20), we conclude that

$$\left\|\hat{\beta} - \vec{\mathcal{I}}\right\| \leq c_2(\sqrt{K} + 1/2) + O(c_3 K^{3/2} p^{-1}) + O(\sqrt{\frac{K + \log(\delta^{-1})}{m}}),$$

with probability at least $1 - \delta$ over the randomness of $m$ random subsets drawn from $\mathcal{D}$, which completes the proof of equation (6). □

**Algorithm 2** Gradient Estimation (`GradEx`) in the case of multi-class classification

**Input:** Subset vector $\mathbf{s}^{(i)}$; precomputed logits $\{f_{W_0}\}$ and gradients $\{\nabla_W f_{W_0}\}$; test dataset $T_{\text{test}}$ and test loss $\ell$
**Requires:** $\ell_2$-regularization $\lambda > 0$; random projection matrix $P \in \mathbb{R}^{k \times d}$
**Output:** Estimated performance on the test task

Construct the sample subset $S^{(i)}_{\text{train}} = \{z_j = (x_j, y_j)\}_{j=1}^{|S|}$ from tasks selected in $\mathbf{s}^{(i)}$
/* *Step 1: Project gradients and solve a low-dimensional regression problem* */
Compute projected gradients for all samples in the subset: $g_j = P\nabla f_{W_0}(x_j)$
Define the low-dimensional objective over $Z \in \mathbb{R}^k$:

$$\tilde{L}(Z) = \sum_{j=1}^{|S|} \ell_{\text{CE}}\left(f_{W_0}(x_j) + \langle g_j, Z\rangle, y_j\right) + \frac{\lambda}{2}\|Z\|_2^2$$

Compute the optimal low-dimensional regression by minimizing this convex objective:

$$Z^\star = \arg\min_{Z \in \mathbb{R}^k} \tilde{L}(Z)$$

/* *Step 2: Estimate test performance* */
$y^{(i)} \leftarrow 0$
**for** each $z = (x, y) \in T_{\text{test}}$ **do**
    Project the test gradient: $\tilde{g} = P\nabla f_{W_0}(x)$
    Approximate the test logits using the low-dimensional projections:

$$\hat{f}(x) = f_{W_0}(x) + \langle \tilde{g}, Z^\star \rangle$$

    Accumulate the loss: $y^{(i)} \leftarrow y^{(i)} + \ell(\hat{f}(x), y)$
**end for**
$y^{(i)} \leftarrow y^{(i)}/|T_{\text{test}}|$ ▷ Normalize the loss over the test dataset
**return** $y^{(i)}$

### A.3 Gradient estimation algorithms

**Gaussian random projection.** This method directly addresses the high dimensionality of the gradient by projecting it into a low-dimensional subspace. We employ a random matrix $P \in \mathbb{R}^{k \times d}$ (where $k \ll d$) to map the gradient $\nabla f_W(x) \in \mathbb{R}^d$ to a compressed representation

$$\nabla \tilde{f}_W(x) = P\nabla f_W(x) \in \mathbb{R}^k.$$

Motivated by the Johnson-Lindenstrauss Lemma, which guarantees that pairwise distances are approximately preserved, we then solve the perturbation objective using these low-dimensional features. The optimization is performed over a $k$-dimensional vector, making the problem tractable and independent of the original parameter count $d$.

**Gradient estimation for in-context learning.** For our experiments on the ICL task, we define the loss function and compute gradients in the embedding space. This approach enables us to avoid perturbation-based objective optimization methods, as we already know the difference in embeddings. Thus, we can directly perform a first-order estimation of the logit outputs for different prompts based on the gap between their embeddings. Consider the model output of a prompt subset $S$ on an input $x$, denoted as $f_W(\phi(S, x))$. Given an anchor prompt $S_0$, the first-order approximation of $f_W$ around the embedding vector $\phi(S_0, x)$ is given by:

$$f_W(\phi(S,x)) = f_W(\phi(S_0,x)) + \langle \nabla_\phi f_W(\phi(S_0,x)), \phi(S,x) - \phi(S_0,x)\rangle + \epsilon_{S,x}. \tag{24}$$

### A.4 First-order approximation error

First, kernel analyses of fine-tuning large transformers show that, in the kernel behavior regime, the model evolution is well-approximated by its first-order Taylor expansion around the pre-trained parameters $\widehat{W}$, such that

$$f_W(x) = f_{W_0}(x) + \langle \nabla f_{W_0}(x), W - W_0\rangle + \epsilon_W(x),$$

Table 5: We report the approximation error on different sizes of models on SST-2 and Coin flip.

| | SST-2 | Coin flip |
|---|---|---|
| DeepSeek-LLM-7B | $7.6_{\pm 4.5} \times 10^{-3}$ | $6.5_{\pm 2.0} \times 10^{-3}$ |
| Llama-3.1-8B | $1.8_{\pm 0.1} \times 10^{-2}$ | $7.5_{\pm 1.2} \times 10^{-3}$ |
| Llama-2-13B | $7.7_{\pm 1.2} \times 10^{-4}$ | $1.5_{\pm 0.7} \times 10^{-3}$ |
| CodeLlama-34B | $7.4_{\pm 2.7} \times 10^{-4}$ | $8.5_{\pm 1.1} \times 10^{-4}$ |

with Taylor expansion residual error term $\epsilon_W(x)$ on $x$. This justifies working in the linearized regime around $W_0$ even for highly nonlinear LLMs.

We show the approximation error on various models with parameters from 7B to 34B. We test on SST-2 and Coin flip datasets. Our findings in Table 5 show that the errors remain negligible across different architectures and model sizes.

Second, assuming the first-order approximation error is small, we show that the gradient estimation algorithm gives accurate estimates. Let $\hat{L}(f_W)$ be the empirical loss, assume that:

- The average Taylor expansion residual is bounded by $\delta$:
$$\mathop{\mathbb{E}}_{(x,y)} \left[ \left| f_W(x,y) - f_{W_0}(x,y) - [\nabla f_{W_0}(x,y)]^\top (W - W_0) \right| \right] \leq \delta,$$
- Gradients at $W_0$ are bounded by $G$, and
- The optimization is restricted to a domain of radius at most $D$, with a random projection that distorts inner products by at most $\epsilon$.

Denote by $\widehat{W}(S)$ the parameter produced by the gradient-based estimator (obtained from regression in the projected gradient feature space, as in our gradient estimation step). One can show that

$$\hat{L}(f_{\widehat{W}(S)}) \leq \min_{W \in D} \hat{L}(f_W) + 2\delta + 4GD\epsilon.$$

Thus, provided that the linearization analysis ensures that the first-order approximation error $\epsilon_W(x)$ (and hence $\delta$) is small, the gradient-based estimator $\widehat{W}(S) = W_0 + Z_s^\star$ achieves training loss within $2\delta + 4GD\epsilon$ of the minimum loss in the search domain. We summarize this discussion into the following proposition.

**Proposition A.2.** *Let $\mathcal{D} \subseteq \mathbb{R}^d$ be a search space whose radius is at most $D$. Suppose the gradient of $f_{W_0}$ at the initialization $W_0$ in the training set is at most $G$ in Euclidean norm. For each task $i = 1, 2, \ldots, n$, let $T_i$ denote the training data. Suppose that for every $i$,*

$$\frac{1}{|T_i|} \sum_{(x,y) \in T_i} \left| f_W(x,y) - f_{W_0}(x,y) - \nabla_W f_{W_0}(x,y)^\top (W - W_0) \right| \leq \delta.$$

*Provided that the random projection dimension $k$ satisfies $k = O\left(\frac{\log N}{\epsilon^2}\right)$, the training loss of $\widehat{W}(S)$ is bounded away from the minimum training loss for any $S \subseteq \{1, 2, \ldots, n\}$ as*

$$\hat{L}(f_{\widehat{W}(S)}) \leq \min_{W \in \mathcal{D}} \hat{L}(f_W) + 2\delta + 4GD\epsilon. \tag{25}$$

The proof is based on the Johnson-Lindenstrauss lemma (Johnson, 1984), which asserts that when $k = O\left(\frac{\log N}{\epsilon^2}\right)$, for any $g_i$ with $\|g_i\| \leq G$ and any $W, W_0$ in $\mathcal{D}$, we have

$$|\langle g_i, W - W_0 \rangle - \langle P g_i, P(W - W_0) \rangle| \leq \epsilon |\langle g_i, W - W_0 \rangle| \leq 2GD\epsilon.$$

The rest of the argument can be completed via simple inequalities.

### A.5 Expressivity of RBF kernels

**Proposition A.3** (Expressivity of RBF kernels)**.** *Let $X = \{0,1\}^K$ and $k$ be the Gaussian RBF kernel. Then:*

*i)* ***Exact interpolation:*** *There exists a function $g \in \mathcal{H}_k$ such that $g(\mathbf{s}) = F(\mathbf{s})$ for all $\mathbf{s} \in X$.*

*ii)* ***Universal approximation:*** *Consequently, for any $\epsilon > 0$, there exists $f \in \mathcal{H}_k$ such that*

$$\max_{\mathbf{s} \in X} \left| F(\mathbf{s}) - f(\mathbf{s}) \right| \leq \epsilon.$$

*Proof.* We first prove Part i). The set $X$ consists of distinct points in $\mathbb{R}^K$. It is a well-established fact that the Gaussian RBF kernel is *strictly positive definite* on any finite set of distinct points (Micchelli, 1984; Schölkopf & Smola, 2002).

Let $m = |X|$ be the number of tasks. We enumerate the elements of $X$ as $\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(m)}$ and let $y_j = F(\mathbf{s}^{(j)})$ be the target values. We define the Gram matrix $K_G \in \mathbb{R}^{m \times m}$ with entries

$$K_{G_{i,j}} = k(\mathbf{s}^{(i)}, \mathbf{s}^{(j)}).$$

Since the kernel is strictly positive definite, the matrix $K$ is strictly positive definite and therefore invertible (full rank). This implies that the linear system $K_G \theta = y$ has a unique solution $\theta \in \mathbb{R}^m$. We construct the function

$$g_\theta(\mathbf{s}) = \sum_{i=1}^{m} \theta_i k(\mathbf{s}^{(i)}, \mathbf{s}).$$

By construction, $g_\theta \in \mathcal{H}_k$ and satisfies $g_\theta(\mathbf{s}^{(j)}) = y_j = F(\mathbf{s}^{(j)})$ for all $j$. This proves Part i).

Part ii) follows immediately from Part i). Since $g$ matches $F$ exactly on $X$, the approximation error is zero, which is always less than or equal to any $\epsilon > 0$. □

The intuition from the above result is that on $\{0,1\}^K$, two subsets that differ on only a few coordinates are close in Hamming distance, and intuitively their performances tend to be similar. The RBF kernel directly reflects this structure:

$$k_{RBF}(\mathbf{s}, \mathbf{s}') = \exp\left( -\frac{1}{2\sigma^2} \|\mathbf{s} - \mathbf{s}'\|_2^2 \right)$$

decays smoothly as the Hamming distance between $\mathbf{s}$ and $\mathbf{s}'$ increases. Thus, kernel regression assigns stronger influence to nearby subsets and weaker influence to distant ones, capturing the local smoothness we observe in practice.

By contrast, polynomial kernels do not encode distance-based locality. A degree-$d$ polynomial kernel expands inputs into global monomials (e.g., $s_i s_j$, $s_i s_j s_k$, ...), so two subsets with small Hamming distance are not necessarily mapped to nearby points in the feature space.

## B Omitted Experiments

We begin by describing additional details that are omitted from the experimental setup in Section 4.1.

**Modular arithmetic tasks.** We consider the following functions with $p = 97$: $a + b \pmod p = c$, and $a^2 + ab + b^2 \pmod p = c$. For each task, we generate a complete dataset by iterating through all possible pairs of $(a, b)$, where $a, b \in \{0, 1, \ldots, p-1\}$. This results in a dataset of $97 \times 97 = 9,409$ unique equations for each function. Each equation is formatted as a sequence of tokens: `a`, `op`, `b`, `=`, `c`, where `op` is `+` for addition and a placeholder token for the quadratic function. We randomly split the whole dataset into a training set comprising $90\%$ of the data and a test set comprising the remaining $10\%$ of the dataset.

We use a standard two-layer decoder-only transformer with embedding dimension $128$ as the classifier for all modular experiments. The model is trained to predict the correct token for the result `c`, given the preceding sequence (`a`, `op`, `b`, `=`). The cross-entropy loss and gradients are computed based on the model's output logits at the final token position.

To support our task attribution analysis, we use a grouped subset sampling strategy that partitions the training data based on the values of both operands, $a$ and $b$. We first partition the range of

possible values for each operand, $\{0, 1, \ldots, 96\}$, into 5 disjoint intervals. Specifically, a training example $(a, b, c)$ is assigned to a group $G_{i,j}$ where $i = \lfloor a/20 \rfloor$ and $j = \lfloor b/20 \rfloor$. This creates a $5 \times 5$ grid of 25 groups, where each group corresponds to a specific rectangular region in the input space. Subsets of the training data are then constructed by selecting specific combinations of these groups. This method enables a fine-grained analysis of how the model learns from different regions of the input space, allowing for the systematic construction of task vectors. We sample $50$ subsets with a sampling ratio $0.9$, and use $40$ subsets as the surrogate training set.

**In-context learning.** We use the Qwen3-8B as the base model and evaluate it on a sentiment classification dataset, SST-2, and a reasoning task, the coin flip. The SST-2 dataset is a binary sentiment classification dataset composed of movie reviews labeled as either positive or negative from the GLUE benchmark. The number of queries is $450$. The Coin-Flip dataset is an arithmetic reasoning task where the model reads a natural language description of a sequence of fair coin flips and must predict the final outcome (heads or tails). The number of queries is $869$. For each dataset, we use the first $50$ candidate demonstrations as the data to be attributed. We sample $200$ subsets, each containing $4$ prompts, and use $80$ of these subsets as the surrogate training set.

**Multi-objective reinforcement learning.** We use MT10 from the Meta-World benchmark (Yu et al., 2020), which consists of 10 diverse robotic manipulation tasks. The agent's observation includes the environment state and a one-hot vector that specifies the current task. A sparse reward for moving the objective to its goal position. The 10 tasks in MT10 are: reach, push, pick-place, door-open, drawer-open, drawer-close, button-press-topdown, peg-insert-side, window-open, and box-open.

For the task attribution evaluation, we use the Soft Actor-Critic (SAC) as the training algorithm, and take the reward of each task as the $F(\mathbf{s})$. We sample $50$ subsets, each containing 7 tasks, and use $40$ subsets as the surrogate training set.

**Hessian-aware training regularizers.** Sharpness-Aware Minimization (SAM) is an optimization method that aims to find parameters lying in flat neighborhoods of the loss landscape, rather than just minimizing the loss at a single point. Concretely, instead of minimizing the empirical loss $L_S(w)$, SAM minimizes the worst-case loss in an $\ell_p$-ball of radius $\rho$ around $w$:

$$\min_w L_S^{\mathrm{SAM}}(w) = \min_w \max_{\|\varepsilon\|_p \le \rho} L_S(w + \varepsilon).$$

Using a first-order Taylor approximation, the inner maximization has approximate solution

$$\hat{\varepsilon}(w) = \rho \frac{\nabla_w L_S(w)}{\|\nabla_w L_S(w)\|_2},$$

for the common case $p = 2$. Each SAM step then computes the gradient of the loss at the perturbed weights, $\nabla_w L_S(w + \hat{\varepsilon}(w))$, and updates $w$ in that direction, which biases training toward flatter minima that empirically generalize better than the sharp minima found by standard SGD.

NSO (Zhang et al., 2024) is an optimizer that explicitly encourages flat minima by minimizing a noise-perturbed loss and thereby regularizing the *trace of the Hessian*. Given an empirical loss $f(W)$ and a zero-mean noise distribution $P$, NSO considers the smoothed objective

$$F(W) := \mathbb{E}_{U \sim P}[f(W + U)],$$

and uses a two-point noise injection scheme: at each step it samples $U \sim P$ and averages gradients at symmetrically perturbed weights,

$$G(W, U) = \frac{1}{2}\left(g(W + U) + g(W - U)\right),$$

which cancels the first-order term in the Taylor expansion and keeps the second-order term

$$\frac{1}{2} U^\top \nabla^2 f(W) U.$$

For isotropic noise with covariance $\Sigma = \mathcal{N}(0, \sigma^2 \,\mathrm{Id})$, this yields

$$F(W) \approx f(W) + \frac{1}{2}\langle \Sigma, \nabla^2 f(W) \rangle.$$

Thus, running gradient descent on $F(\cdot)$ with Gaussian convolution is approximately the same as running gradient descent on $f(\cdot)$ plus a penalty term of $(\sigma^2/2) \cdot \mathrm{Tr}[\nabla^2 f(W)]$.

Table 6: Top positive and negative influential prompt samples identified on the coin-flip task.

| Query | Top positive prompt samples | Top negative prompt samples |
|---|---|---|
| Q: A coin is heads up. Kielmeyer **flips** the coin. Jevgenij **does not flip** the coin. Is the coin still heads up? **(no)** | Q: A coin is heads up. Erdener **flips** the coin. Ismari **does not flip** the coin. Is the coin still heads up? **(no)** | Q: A coin is heads up. Pachl *flips* the coin. Lissett *flips* the coin. Is the coin still heads up? *(yes)* |
| Q: A coin is heads up. Bonnitta **does not flip** the coin. Ellise **does not flip** the coin. Is the coin still heads up? **(yes)** | Q: A coin is heads up. Brittingham **does not flip** the coin. Alilet **does not flip** the coin. Is the coin still heads up? **(yes)** | Q: A coin is heads up. Kimyetta *flips* the coin. Raynel **does not** flip the coin. Is the coin still heads up? *(no)* |
| Q: A coin is heads up. Roeland **does not flip** the coin. Joeliz **flips** the coin. Is the coin still heads up? **(no)** | Q: A coin is heads up. Kulju **does not flip** the coin. Afrodisio **flips** the coin. Is the coin still heads up? **(no)** | Q: A coin is heads up. Pachl *flips* the coin. Lissett **flips** the coin. Is the coin still heads up? *(yes)* |

**Attribution baselines.** Influence functions (Koh & Liang, 2017) are proposed to measure the influence of individual samples, which can also be adapted to estimate the influence of entire tasks by considering a task-weighted loss function. The influence of infinitesimally up-weighting task $T_k$ on the test performance is given by:

$$\mathcal{I}_k(\mathbf{s}) = \left[\nabla_W F(\mathbf{s})\right]^\top \left[\nabla^2_W L(f_W, \mathbf{s})\right]^{-1} \nabla_W \left( \frac{s_k}{\sum_{j=1}^K s_j} \ell_k(f_W, T_k) \right).$$

Here, $H = \nabla^2_W L(f_W, \mathbf{s})$ is the Hessian of the total training loss. In our implementation, we approximate the inverse Hessian-vector product $H^{-1}v$ (where $v$ is the gradient of the task-specific loss term) by solving a Lasso regression problem, which avoids direct matrix inversion.

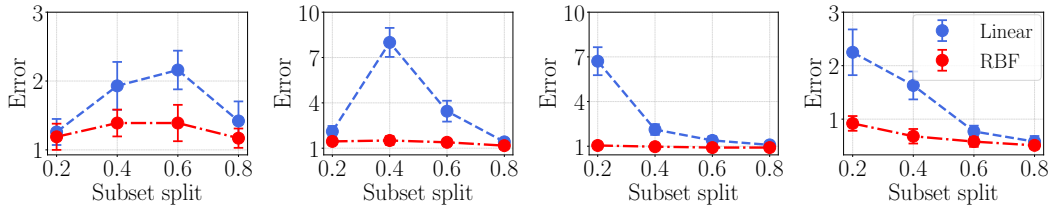
TracIn (Pruthi et al., 2020) traces the influence of training data through the optimization trajectory, avoiding Hessian computation. We adapt this method from the sample level to the task level by measuring the correlation between task gradients over the course of training. The influence of a training task $T_k$ on a test task $T_{\text{test}}$ is approximated by summing the dot products of their respective loss gradients across various training checkpoints:

$$\text{TracIn}(T_k, T_{\text{test}}) = \sum_{t=1}^{T} \eta_t [\nabla f_{\text{test}}]^\top \nabla \ell_k(f_{W_t}, T_k),$$

where $W_t$ are the model parameters and $\eta_t$ is the learning rate at checkpoint $t$.

TRAK (Park et al., 2023) offers an efficient algorithm for data attribution by linearizing the model and using random projections. We adapt it to attribute influence at the task level. The core idea is to represent each task by an average of its constituent samples' projected gradients. Specifically, for each sample $z$ in a task, a feature vector is computed from the gradient of a model output function $f_W$. To get a feature vector for an entire task $T_k$, we average these features over all its samples. This task-level feature is then projected into a low-dimensional space using a random matrix $\mathbf{P}$. The attribution score for the test task $T_{\text{test}}$ is then computed as: $\phi(T_{\text{test}})^\top (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{Q}$. Here, $\phi(T_{\text{test}}) \in \mathbb{R}^k$ is the projected feature vector for the test task, $\mathbf{\Phi}$ is the $K \times k$ matrix of stacked projected features for the $K$ training tasks, and $\mathbf{Q}$ is a $K \times K$ diagonal weighting matrix. The final scores are averaged over an ensemble of models to ensure robustness.

Linear surrogate models (Ilyas et al., 2022; Li et al., 2023) learn a linear mapping from data weights to model predictions. Given a set of models trained on different data subsets, this approach fits a linear function $f : \{0,1\}^n \to \mathbb{R}$ that predicts test performance from subset indicators. The coefficients of this linear model serve as attribution scores, capturing the marginal contribution of each training example across multiple training runs.




(a) Modular addition (b) Modular quadratic (c) SST-2 (d) Coin flip

Figure 4: We investigated how the size of the surrogate training split affects the residual error of the linear and kernel models. We observed that across a range of training splits from 20% to 80%, the kernel model consistently yields a lower residual error than the linear model.

### B.1 Omitted experimental results

**Qualitative results.** In Table 6, we present the most positively and negatively prompt samples obtained by KernelSM on the Coin flip dataset. We use bold to show information that is the same as the query, and italics for information that is different from the query.

**Correlation between surrogate models and influence functions.** We designed an experiment using an $\ell_2$-regularized logistic regression model (with a $10^{-2}$ $\ell_2$ penalty) on the Wisconsin Breast Cancer dataset. The data was standardized and split into a training set of 455 samples and a test set. Our analysis focused on quantifying the influence of each training point on the loss of a single, randomly selected test point.

We computed three distinct valuation scores for every training point: the ground-truth leave-one-out (LOO) score, the first-order influence function (IF) approximation, and the coefficients from a linear surrogate model. The LOO scores were obtained by exhaustively retraining the model from scratch (cold-start) after removing each training point individually and recording the change in test loss. The IF scores were calculated as a first-order approximation using the Hessian of the full training loss, stabilized with a $10^{-3}$ damping term. To build the surrogate model, we generated 1000 subsets, each containing 430 training points, and retrained a model on each to measure the resulting test loss change. These loss changes were then used as targets in a final linear regression, whose coefficients, solved via ordinary least squares (OLS), provided the surrogate model scores.

**Nonlinear numerical simulation.** We consider a binary classification task in a feature space $\mathcal{X} \in \mathbb{R}^2$ with labels $\mathcal{Y} \in \{0, 1\}$. The base learning algorithm, denoted by $\mathcal{A}$, is a two-layer MLP. The network architecture consists of an input layer, two hidden layers with 16 neurons each and ReLU activation functions, and a final output layer with a softmax activation.

To analyze the influence of different training samples, we construct a series of related but distinct training subsets. We first define a small, fixed set of $N$ anchor data points, $S_{\text{anchor}} = \{(x_i, y_i)_{i=1}^N\}$. Next, we define a candidate set of $K$ additional data points, $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, which are sampled along the path that traverses a critical region near this boundary. The experiment then focuses on a series of $K$ training subsets, where each subset $S_j$ is formed by combining the fixed anchor set with exactly one candidate point from $\mathcal{C}$: $S_j = S_{\text{anchor}} \cup \{(c_j, y_c)\}, \quad \text{for } j = 1, \ldots, K$. We select a fixed test point $x_{\text{test}} \in \mathcal{X}$ whose prediction is sensitive to the location of the decision boundary. The goal is to attribute the model's prediction on $x_{\text{test}}$ to the choice of the candidate point $c_j$.

### B.2 Ablation studies

Kernel methods often require a higher sample complexity to converge to an optimal solution compared to linear models. We conduct an ablation study to investigate this trade-off. We compared the performance of the kernel surrogate model with that of the linear surrogate model, varying the size of the data subset used for their construction, sampling from 20% to 80% of the total subsets. The results in Figure 4 show that the kernel surrogate model consistently outperforms the linear surrogate model across all tested subset sizes. Notably, even when the number of samples was small (e.g., at the 20% subset level), the kernel method still demonstrated a better performance.

Table 7: We compare the performance of KERNELSM under different $\lambda$ and $\gamma$.

| | $\lambda = 1$ | $\lambda = 10^{-1}$ | $\lambda = 10^{-2}$ | $\lambda = 10^{-3}$ | $\gamma = 10^{-1}$ | $\gamma = 10^{-2}$ | $\gamma = 10^{-3}$ | $\gamma = 10^{-4}$ | $\gamma = 10^{-5}$ |
|---|---|---|---|---|---|---|---|---|---|
| Error | $0.81_{\pm 0.01}$ | $0.65_{\pm 0.01}$ | $0.73_{\pm 0.04}$ | $0.86_{\pm 0.10}$ | $0.65_{\pm 0.01}$ | $0.66_{\pm 0.01}$ | $0.87_{\pm 0.01}$ | $1.01_{\pm 0.03}$ | $1.02_{\pm 0.03}$ |
| LDS | $0.52_{\pm 0.03}$ | $0.54_{\pm 0.01}$ | $0.51_{\pm 0.02}$ | $0.51_{\pm 0.06}$ | $0.54_{\pm 0.01}$ | $0.53_{\pm 0.02}$ | $0.52_{\pm 0.04}$ | $0.51_{\pm 0.02}$ | $0.51_{\pm 0.04}$ |

**Hyperparameters analysis.** We evaluate the sensitivity of our approach to different hyperparameter settings in the in-context learning task in Table 7. Specifically, we vary $\lambda$ from $10^{-3}$ to 1 and $\gamma$ from $10^{-5}$ to $10^{-1}$. We find that our approach is relatively robust to changes in both $\lambda$ and $\gamma$, exhibiting stable performance across the entire range.

In our experiments, we set $\lambda = 10^{-1}$ and $\gamma = 1/n$ as the default configuration, where $n$ denotes the number of tasks.

**Comparison of kernels.** We use the CIFAR-10 dataset and the ResNet-9 network classifier for our experiments. The dataset consists of 60,000 $32 \times 32$ color images in ten classes, with 6,000 images per class. We utilize the standard training set of 50,000 images. To align with the scope of our task attribution methodology, we partition the 50,000 training samples into 50 disjoint tasks. Each task is constructed by randomly sampling 1,000 data points from the training set without replacement. This results in 50 distinct tasks, which form the basis of our analysis. For the training of our surrogate model, we randomly select 30 of these 50 tasks.

We evaluate the performance of different kernels. Let $\mathbf{s}^{(a)}$ and $\mathbf{s}^{(b)}$ denote different subset indices. The kernels evaluated in our study are:

- Polynomial kernel (Degree-1, 2, 3): The general form of the polynomial kernel is
$$k(\mathbf{s}^{(a)}, \mathbf{s}^{(b)}) = \big((\mathbf{s}^{(a)})^\top \mathbf{s}^{(b)} + c\big)^d.$$
In our experiments, we set the constant $c = 0$ and test for degrees $d \in \{1, 2, 3\}$.
- Radial Basis Function (RBF) kernel: The RBF kernel is defined by the following equation:
$$k(\mathbf{s}^{(a)}, \mathbf{s}^{(b)}) = \exp\big(-\gamma \|\mathbf{s}^{(a)} - \mathbf{s}^{(b)}\|^2\big),$$
where $\gamma$ is a parameter that controls the smoothness of the kernel function relative to the distance between $\mathbf{s}^{(a)}$ and $\mathbf{s}^{(b)}$.

In Table 8, we find that the linear surrogate model and degree-1 kernel have a large residual error. The degree-2 kernel achieves the lowest residual error among the three polynomial kernels. The RBF kernel achieves the lowest residual error among all the kernels.

### B.3 EXTENSIONS AND DISCUSSIONS

Model-agnostic meta-learning (MAML) seeks to learn a model initialization that is optimized for rapid adaptation to new tasks, typically with only a few gradient steps. Our work on task attribution, which measures a model's sensitivity to its training data, is connected to this goal. The error of first-order attribution methods is directly governed by the Hessian. For attribution, a large Hessian signifies a highly curved, non-linear performance landscape where linear approximations are unreliable, leading to significant attribution error. In contrast, MAML leverages this same curvature as a signal. The meta-gradient update in MAML involves differentiating through an inner-loop gradient step, a calculation that explicitly depends on the Hessian of the inner-loop loss with respect to the model parameters. There also exists a mathematical parallel between the derivation of influence functions and the formulation of implicit MAML (iMAML) (Rajeswaran et al., 2019). Influence functions can be derived by applying the implicit function theorem to the first-order optimality condition of the perturbed loss, yielding an expression for the parameter change that involves the inverse Hessian. Analogously, iMAML uses the implicit function theorem to derive an analytical expression for the meta-gradient that depends only on the solution of the inner optimization, not the path taken to reach it.

Multi-group learning aims to train a single predictor that performs robustly across a predefined set of subgroups, addressing the common failure mode where high average accuracy masks poor performance on critical sub-populations (Deng & Hsu, 2024). The influence of task $k$ on the performance

Table 8: We investigate the surrogate model performance with different kernels. We run each experiment with five random seeds to report the standard deviations.

| Residual error | Linear model | Degree-1 | Degree-2 | Degree-3 | RBF |
|---|---|---|---|---|---|
| CIFAR-10 | $4.4_{\pm 0.9}$ | $3.8_{\pm 0.8}$ | $1.6_{\pm 0.1}$ | $2.7_{\pm 0.2}$ | $1.0_{\pm 0.0}$ |
| Modular arithmetic | $4.6_{\pm 1.3}$ | $2.2_{\pm 0.5}$ | $1.7_{\pm 0.4}$ | $3.3_{\pm 0.7}$ | $1.5_{\pm 0.4}$ |
| In-context learning | $0.8_{\pm 0.2}$ | $0.6_{\pm 0.1}$ | $0.5_{\pm 0.1}$ | $0.5_{\pm 0.1}$ | $0.4_{\pm 0.1}$ |
| Multi-objective RL | $0.2_{\pm 0.1}$ | $0.2_{\pm 0.1}$ | $0.1_{\pm 0.1}$ | $0.1_{\pm 0.1}$ | $0.1_{\pm 0.1}$ |

of a model evaluated on task $j$ can be directly interpreted as the marginal contribution of group $k$'s training data to the model's performance on group $j$. A positive influence value for a loss-based metric provides a clear signal of negative transfer. Standard influence functions, being first-order approximations, capture pairwise, additive effects and struggle to model more complex issues. For example, a model may perform poorly on an intersectional group (e.g., women of a specific race) due to higher-order interactions between the data subgroups that linear methods cannot detect. By using a non-linear RBF kernel, our model learns a global function that captures combinatorial effects between groups. This can also inform the design of robust learning algorithms. For example, the MGL-Tree algorithm for hierarchical groups decides whether to use a general parent-level predictor or a specialized child-level predictor by comparing their empirical risks. Our method could provide a more principled, causal signal to guide this choice.

**Discussions.** We now discuss this work with several related literature. There is an earlier line of work called LIME (Ribeiro et al., 2016), which addresses the question of explaining a single prediction at the feature level by fitting a surrogate that maps feature perturbations of a specific input to changes in the model output.

Our work instead focuses on training-data attribution by modeling how changing the weights of training examples affects the model's behavior. Although both approaches rely on local surrogate modeling, they operate on different objects. As a result, our work is complementary to this important line of work on LIME.

In terms of what's new in this paper relative to this earlier line of work, our first observation is to connect linear surrogate models to influence functions; this is shown in Section 3.1, where we find the Ordinary Least Squares (OLS) regressor is approximately equal to influence functions (Koh & Liang, 2017). This observation partially explains why linear surrogate models can perform well empirically (Ilyas et al., 2022).

At the same time, we identify a limitation of linear surrogate models in the context of task/data attribution, which motivates our kernel-based surrogate modeling. Kernels offer a universal function approximation framework, enabling the surrogate to capture nonlinear dependencies between training data and model behavior. To the best of our knowledge, no prior work in data attribution uses a kernel-based surrogate.

In the context of task attribution, kernel surrogate models are challenging to estimate due to the need for repeated training. Our second contribution is an efficient gradient-based estimator for fitting the kernel surrogate model. The computational cost consists of three parts: pre-computing $f_{\hat{W}}$, gradient estimation, and surrogate fitting. Except for the pre-computation step, which only involves computing gradients once on the training data, all the remaining computations can be executed quickly on CPUs.

As another remark, our in-context learning focuses on text classification or linear regression cases where the ordering of the examples does not affect the outcome. This setting builds on the prior work of Min et al. (2022), who found that for many ICL classification tasks, model outputs are often not sensitive to permutations of the context examples. We verify this in our own experiments by evaluating the model under multiple random permutations of the same demonstration set: the standard deviation across different orders is only about $11\%$ of the mean loss, indicating that order effects are relatively minor in our setting. This justifies our choice to model only which demonstrations are included, rather than their specific ordering.

Beyond this setup, we agree that there are important scenarios, such as math reasoning with chain-of-thought style prompts, where the order of demonstrations plays a significantly larger role. Our framework can be naturally extended to handle such cases. Concretely, we can treat different orderings of the same demonstration set as different inputs to the surrogate model. For each permutation $\pi$ of a fixed set of examples, we would evaluate the corresponding performance $F(\mathbf{s}, \pi)$ and fit the kernel surrogate on $(\mathbf{s}, \pi)$, thereby assigning different scores to different orderings. In this way, our method can explicitly model and attribute order-sensitive effects when they are significant. This would be an interesting question for future work.